OULUN SEUDUN
AMMATTIKORKEAKOULU

Pauli Marjakangas

HUOLTOLOKIJÄRJESTELMÄ

HUOLTOLOKIJÄRJESTELMÄ

Pauli Marjakangas
Opinnäytetyö
Syksy 2011
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikka, Ohjelmistotuotanto

Tekijä(t): Pauli Marjakangas
Opinnäytetyön nimi: Huoltolokijärjestelmä
Työn ohjaaja(t): Ensio Sieppi
Työn valmistumislukukausi ja -vuosi: Syksy 2011 Sivumäärä: *47 + 2*

Opinnäytetyön tavoitteena oli sähköisen huoltolokijärjestelmän kehittäminen pieniä kotimaisia elektromekaniikan valmistustoimintaa harjoittavia yrityksiä varten. Tavoitteena oli kattaa myös sellaisten yritysten tarpeet, joilla mahdollisesti on käytössään ISO 9001 -standardin mukainen laadunhallintajärjestelmä ja ISO 14001 -standardin mukainen ympäristöjärjestelmä. Kehitystyötä ei tehty minkään tietyn yrityksen tarpeisiin perustuen, vaan tavoite oli saada aikaiseksi yleiskäyttöinen järjestelmä, joka myöhemmin voitaisiin kaupallistaa niin haluttaessa. Kaupallistaminen ei kuulunut työn tavoitteisiin, eikä järjestelmälle ollut pilottiasiakasta. Työn tavoiteasetannan selkeyttämiseksi suoritettiin suppea internettutkimus, jonka avulla kartoitettiin markkinoilta vastaavaan tarkoitukseen löytyvien ratkaisujen tarjonta ja sisältö.

Varsinainen kehitystyö suoritettiin 4-vaiheisena ohjelmistotuotantoprojektina, jonka suunniteltu kesto oli 11 viikkoa. Projektin vaiheita olivat määrittely-, suunnittelu-, toteutus- ja lopetusvaihe. Määrittelyvaiheen aikana asiakasvaatimusten pohjalta hahmoteltiin järjestelmä, joka koostui neljästä erillisestä osasta: 1. Tietokanta (MySql), 2. Ylläpito-ohjelma (C# Windows Forms Application), 3. Asiakasohjelma (C# Windows Forms Application), 4. Etäkäyttöliittymä (HTML/PHP). Järjestelmäosa numero 4 oli ehdollinen ja se päätettiin toteutusvaiheen aikana jättää pois. Sen sisältämä toiminnallisuus sisällytettiin ylläpito-ohjelmaan.

Kehitystyön tuloksena syntyi järjestelmä, jonka avulla asiakas kykenee hallinnoimaan erilaisten volyymituotannon testauksessa käytettävien laitteistojen ja laitteiden ylläpito-, kalibrointi- ja korjaustoimintaa. Järjestelmä tuottaa tarvittavat palvelut, joita tarvitaan ennakoivien huoltotoimien määrittämiseen ja seuraamiseen sekä korjaustoimien tilastointiin. Järjestelmää voidaan käyttää apuna myös laitehallinnassa laajemmassa merkityksessä. Sen avulla voidaan yksilöidä jokainen yksittäinen kohde ja sille määritetyt tiedot. Tätä palvelua voidaan hyödyntää esimerkiksi ISO-auditointien yhteydessä, jolloin laitteiden sijainti ja huoltotilanne tulee voida todentaa. Laitehallintaan liittyen järjestelmään lisättiin myös toiminnallisuus, joka kerää laitteistojen käyttöaika- ja tilatietoja myöhempää tarkastelua varten. Kaikki järjestelmään taltioidut tapahtumat sisältävät aikaleiman, joten ne ovat jäljitettäviä. Järjestelmän tietojen säilyvyyden turvaamiseksi ylläpito-ohjelmaan luotiin toiminnallisuus, jonka avulla käyttäjille voidaan luoda yksilölliset käyttäjätunnukset ja niihin liittyvät käyttöoikeudet sekä toiminnallisuus tietokannan varmuuskopioiden luomiseen ja palauttamiseen.

Työ valmistui alkuperäisen suunnitelman mukaisesti ennalta asetetun aikataulun puitteissa. Vaikka työn sisältöön tehtiin kehitystyön edetessä joitain muutoksia, voitaneen sanoa, että alkuperäiset työlle asetetut tavoitteet ja vaatimukset toteutuivat.

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Engineering

Author(s): Pauli Marjakangas
Title of thesis: Maintenance log management system
Supervisor(s): Ensio Sieppi
Term and year when the thesis was submitted: Autumn 2011 Number of pages: 47 + 2

The objective of this thesis work was to develop an electrical maintenace log management system for small domestic companies who work in a field of electromechanical device manufacturing. Companies which have ISO 9001 compatible quality management system and ISO 14001 compatible environmental management system were also in the scope of the work. Maintenance log system was not developed specificly for the use of any partical company. The aim was to make it general purpose and so it could be commercialized later if so desired. Commercializing system and piloting it with real customer was out the scope of the work. Anyhow, to bring more focus to target setting a brief internet study was completed. Study gave information about what was already available on the market and what kind of functionality those are offering for the customer.

Development work was done as a 4 phase sofware engineering project. Estimated duration for the project was 11 weeks. Project included definition, design, development and closing phases. Maintenace log system parts were outlined in defenition phase based on customer requirement. At this phase there was 4 parts: 1. Database (MySql), 2. Maintenance management program (C# Windows Forms Application), 3. Customer program (C# Windows Forms Application), Interface for remote users (HTTP / PHP). System part number 4 was later decided to be left out of the scope as it was only optional. Planned fuctionality for it was built into system part 2.

As a result of development actions a system was created offering needed services for the user to manage service, repair and calibration actions to be done for serveral different kind of test equipment used in volume production testing. System offers services needed to set and follow preventine maintenace action and keep record about repair actions done. System can be used also for device management as it possesses needed functionality to give unic identification for each device and related information for those. This service can be used for example to help locating instruments from the factory floor and to verify maintenace statuses of those for auditors in case ISO audit is in hand. Also, related to device management, there was functionality added into system enabling it to capture operation time and operation statuses from test systems for later analysis. All information related to events stored into system's database include time stamp offering full traceabilty. Funtionality used for granting user rights with unic username and password and functionality used for backing up and restoring the database was implemented into manintenace program to ensure the safety of system data.

The work was completed according to plans and in given time. Even though there were changes made to the content of work it can be said that given objectives and requirements were met.

SISÄLTÖ

1 JOHDANTO

Sähköistä toiminnallisuutta sisältävien sarjatuotantovalmisteisten laitteiden valmistusprosessi sisältää tyypillisesti useita erilaisia testausvaiheita. Testauksen tarkoituksena on varmistaa kyseiseen vaiheeseen mennessä tehdyn kokoonpanotyön ja valmistuksessa käytettyjen komponenttien oikeellisuus.

Testauksessa käytetyt laitteistot ovat usein käyttötarkoitukseen räätälöityjä järjestelmiä, jotka koostuvat erilaisista mittalaitteista, virtalähteistä, ohjaimista ja mekaniikasta. Yksinkertaisimmillaan laitteisto voi olla pelkkä yksittäinen mittalaite, joka liitetään mitattavaan kohteeseen manuaalisesti, esimerkiksi mittakaapeleiden avulla. Kompleksisemmat laitteistot sisältävät useita instrumentteja ja ovat joko osin, tai kokonaan automatisoituja.

Tuotannon testauslaitteistot ovat valmistustoimintaa harjoittavalle yritykselle mittava investointi ja siksi niistä halutaan pitää hyvää huolta, jotta käyttöaste pysyisi korkeana ja käyttöikä mahdollisimman pitkänä. Tuotannon testaukselle on lisäksi olemassa tarkat laadulliset vaatimukset, joita asettavat tuotteen omat toiminnalliset vaatimusmääritykset sekä erilaiset standardit, kuten ISO9001 (luku 2.1) ja ISO14001 (luku 2.2).

Työn tavoitteena oli suunnitella ja toteuttaa huoltolokijärjestelmä, jonka avulla voidaan hallinnoida yllämainitun kaltaisten tuotannon testilaitteistojen ja laitteiden kaikkia huolto-, kalibrointi- ja korjaustoimia. Huoltolokijärjestelmän kohdeasiakasryhmäksi valittiin pienet kotimaiset elektromekaniikan valmistustoimintaa harjoittavat yritykset. Työssä tuli huomioida myös sellaiset yritykset, jotka omassa toiminnassaan noudattavat yllämainittujen ISO-standardien vaatimuksia ja ohjeita.

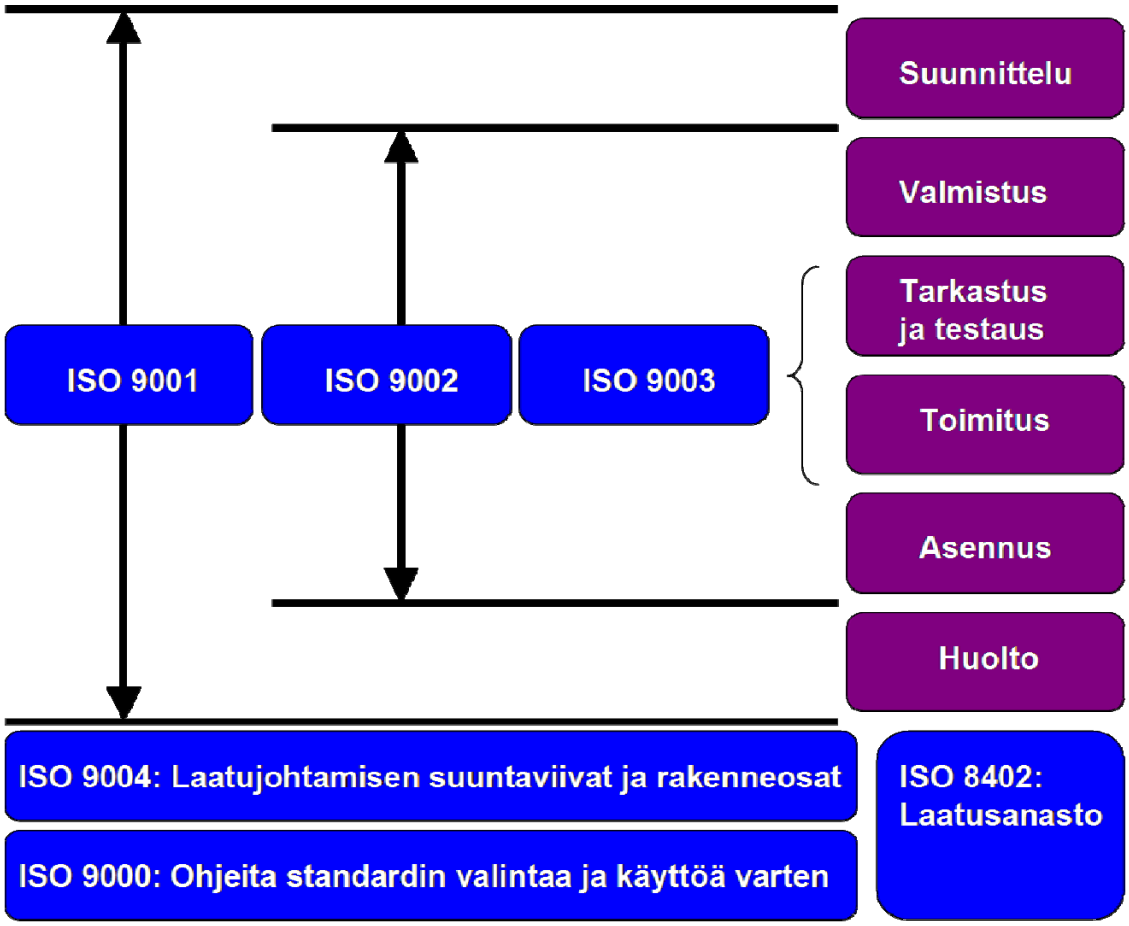
.

2 ISO-STANDARDIT

ISO, eli the International Organization for Standardization, on maailmanlaajuinen kansallisten standardoimisjärjestöjen liitto. ISO-starndardit valmistellaan yleensä ISO:n teknisissä komiteoissa. Jokaisella jäsenellä, joka on kiinnostunut teknisen komitean tehtäväalueella olevasta asiasta, on oikeus olla edustettuna komiteassa. Myös kansainväliset ISO:n kanssa yhteistyössä olevat viranomais- ja muut organisaatiot osallistuvat työhön. ISO työskentelee läheisessä yhteistyössä IEC:n (the Electrotechnical Comission) kanssa kaikissa sähkötekniseen standardisointiin liittyvissä asioissa. Kansainväliset standardit laaditaan ISO/IEC:n sääntöjä (ISO/IEC Directives, Part 3) noudattaen. Teknisten komiteoiden hyväksymät kansainväliset standardiehdotukset jaetaan ISO:n jäsenille äänestystä varten. Kansainvälisen standardin hyväksyminen vaatii, että vähintään 75 % äänestäneistä hyväksyy ehdotuksen. (1, s. 8.)

2.1 ISO 9001 -standardi

ISO 9000 -standardisarja sisältää useita toisiaan täydentäviä osia, kuten ISO 9000, joka antaa ohjeita standardiosien valintaa ja käyttöä varten, tai ISO 9004, joka antaa suuntaviivat organisaation johdolle laatujärjestelmän rakenneosista. Sarjan kolme pääosaa ovat ISO 9001, ISO 9002 ja ISO 9003 (kuva 1).



KUVA 1. Kolmen ISO 9000 -päästandardin kattavuusalueet sekä yleiset standardiosat

ISO 9001 -standardi määrittelee vaatimukset, joiden mukaisesti yritys voi luoda itselleen oman laadunhallintajärjestelmän. Kaikki standardin sisältämät vaatimukset ovat yleisluontoisia, ja on tarkoitus, että ne soveltuvat kaikenlaisille yrityksille sekä organisaatioille koosta ja tuotettavista tuotteista riippumatta. (1, s. 14.)

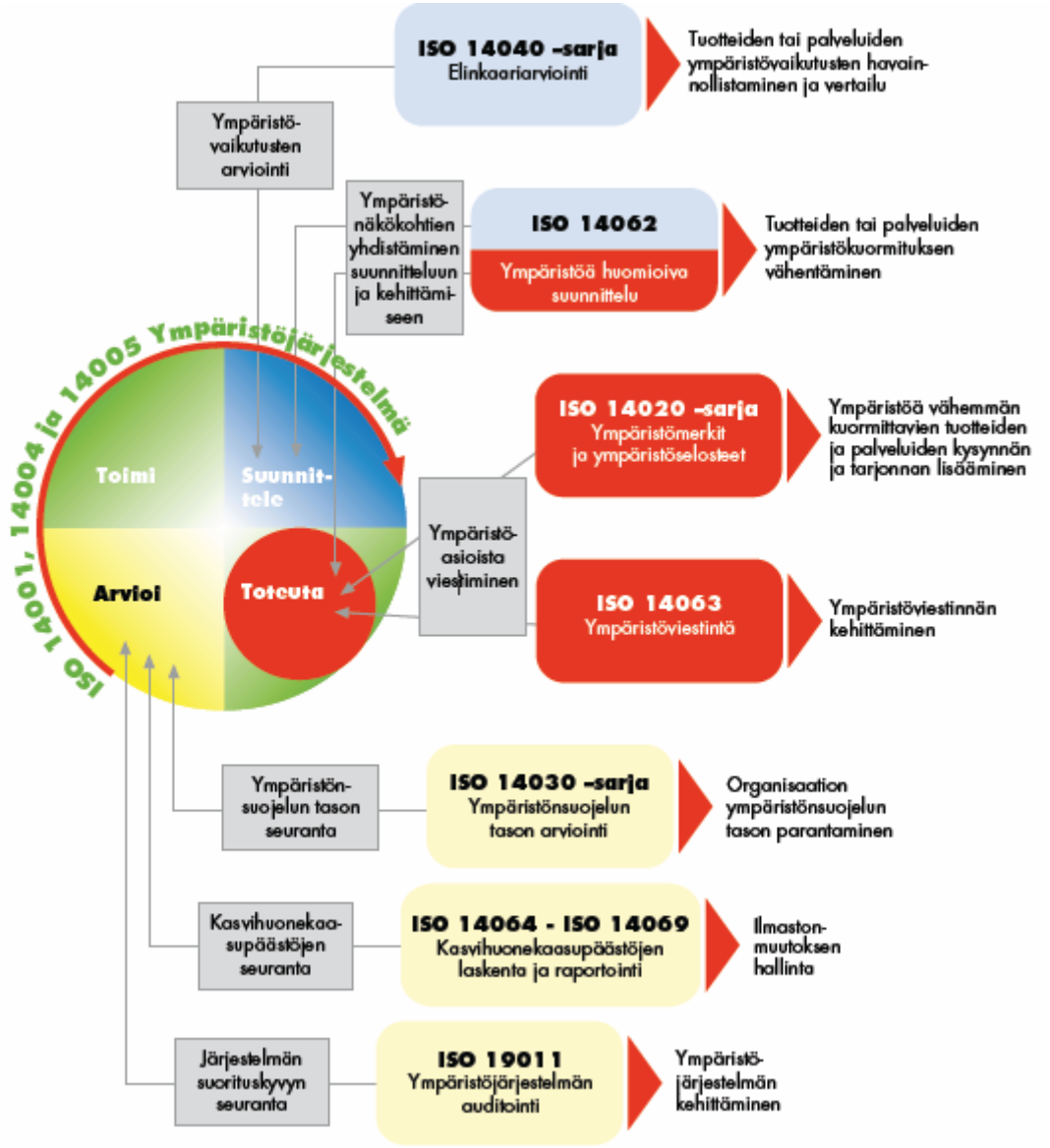
Laatujärjestelmän luomisen taustalla on yleensä yrityksen tavoite parantaa oman organisaationsa toimintaa ja tuottavuutta. Tavoitteena voi olla myös esimerkiksi kilpailukyvyn lisääminen markkinaosuuden säilyttämiseksi tai kasvattamiseksi. Myös asiakkaan asettamat tai soveltuvat lakisääteiset vaatimukset voivat olla syynä siihen, että yritys katsoo tarpeelliseksi oman laatujärjestelmän kehittämisen ja käyttöönoton. Laatujärjestelmän auditoi ja sertifioi ulkopuolinen akkreditoitu sertifiointielin. (2.)

Tässä työssä ISO 9001 -standardin vaatimuksiin viitataan pääasiassa seuranta- ja mittauslaitteiden ohjaukseen liittyviltä osin alla olevien määrittelyiden mukaisesti (lähinnä kohdat a ja c):

> "Organisaation tulee määrittää suoritettavat seurannat ja mittaukset sekä seuranta- ja mittauslaitteet, joita tarvitaan osoittamaan tuotteen täyttävän määritetyt vaatimukset. Organisaation tulee myös luoda prosessit sen varmistamiseksi, että seuranta ja mittaukset voidaan suorittaa, ja että ne myös tehdään siten, että ne täyttävät niille asetetut vaatimukset. Kun kelvollisten tulosten varmistaminen on välttämätöntä, mittauslaitteet tulee
>
> a) kalibroida tai todentaa, joko määräajoin tai ennen käyttöä. Kalibrointi tulee suorittaa jäljitettävissä olevin kansainvälisin, tai kansallisiin mittanormaaleihin verraten. Jos tällaisia mittanormaaleja ei ole, kalibroinnin tai tarkastuksen peruste tulee tallentaa
>
> b) säätää tai uudelleensäätää tarpeen mukaan
>
> c) tunnistaa, jotta kalibroinnin tila voidaan määrittää
>
> d) suojata vahingoittumiselta ja turmeltumiselta käsittelyn, huollon ja varastoinnin aikana." (1, s. 32, 34.)

2.2 ISO 14001

ISO 9000 -standardisarjan tavoin ISO 14000 -standardisarja sisältää useita toisiaan täydentäviä osia. Osien käyttötarkoitus käy ilmi kuvasta 2.



KUVA 2. ISO 14000 -standardien käyttö (9, s. 4.)

ISO 14001 -standardi määrittelee ympäristöjärjestelmää koskevat vaatimukset. Näiden vaatimusten pohjalta yritys voi luoda itselleen oman ympäristöasioita koskevan politiikan ja tavoitteet, joissa huomioidaan vallitseva lainsäädäntö ja tunnistetaan omien toimien ympäristövaikutukset. Standardia sovelletaan niihin ympäristönäkökohtiin, joita organisaatio voi hallita ja joihin sen voidaan olettaa voivan vaikuttaa. Standardi ei suoraan aseta erityisiä ympäristösuojelun tason kriteerejä, vaan kaikki vaatimukset on asetettu siten, että ne voitaisiin sisällyttää kaiken tyyppisiin ympäristöohjelmiin. (3, s. 12.)

Oman sertifioidun ympäristöjärjestelmän hankkimisella yritys kykenee osoittamaan, että se on selvillä lakisääteisistä velvoitteista, sitoutunut

noudattamaan niitä ja on näin yhteiskunnallisesti vastuuntuntoinen toimija (4). Ympäristöjärjestelmä voi olla esimerkiksi lisänä edellisessä luvussa kuvatulle laadunhallintajärjestelmälle ja voi siten toimia myös kilpailuvalttina markkinoilla ympäristönäkökohtia arvostavan asiakkaan silmissä.

3 KÄYTETYT OHJELMISTOT JA KEHITYSTYÖKALUT

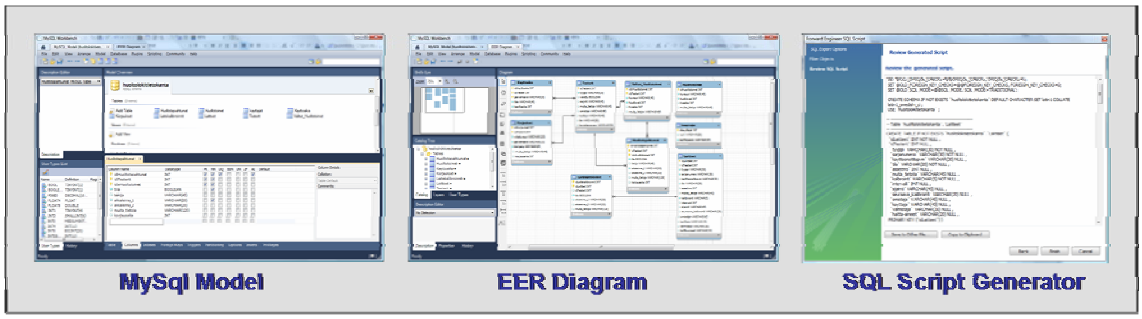
Seuraavissa alaluvuissa esitellään opinnäytetyön tekemisessä käytetyt ohjelmistot ja kehitystyökalut.

3.1 MySql

MySql on maailman suosituin avoimen lähdekoodin tietokantaohjelma, josta on tehty yli 100 miljoonaa kopiota sen historian aikana. Sen suosio perustuu ylivoimaiseen nopeuteen, luotettavuuteen ja helppokäyttöisyyteen. Näiden ominaisuuksien avulla siitä on tullut ensisijainen valinta telekommunikaatio- ja edelläkävijäyritysten IT-johtajille, jos käytössä on Web, Web 2.0, Saas tai ISV, koska se eliminoi tyypilliset ongelmat, jotka liittyvät modernin verkossa olevan tietokannan alhaallaoloaikaan, ylläpidon tarpeeseen ja hallintaan. Monet maailman johtavista organisaatioista käyttävät MySql:ää verkkosivujensa ja ohjelmistojensa jakamiseen tai yritysjärjestelmiensä pyörittämiseen ja säästävät siten aikaa ja rahaa. (5.)

MySql on osa LAMP:tä (Linux, Apache, MySql, PHP/Perl/Python), nopeasti kasvavaa avoimen koodin ohjelmapakettia. Yhä useammat tahot käyttävät kyseistä ohjelmapakettia korvaamaan kalliimpia patentoituja ohjelmia. MySql perustettiin ja kehitettiin ruotsissa 1980-luvulla. Kehitysryhmään kuului kaksi ruotsalaista (David Axmark ja Allan Larsson) ja yksi suomalainen (Michael Widenius). (5.)

Tietokannan suunnitteluun ja toteuttamiseen käytettiin MySql Workbench -kehitystyökalua (kuva 3). MySql Workbench on ohjelma, jonka avulla voidaan suunnitella, kehittää ja hallita tietokantoja.



KUVA 3. Erilaisia suunnittelunäkymiä MySql Workbench -editorissa

3.2 Apache

Apache HTTP -palvelin on avoimen koodin kehitystyön tulos, jossa tavoitteena on ollut luoda luotettava, kaupallista tasoa vastaava HTTP-palvelin (Web), joka on vapaasti saatavilla ja varustettu monipuolisilla toiminnoilla. Palvelimen kehitystyötä ohjaa vapaaehtoisista henkilöistä koostuva ryhmä. Ryhmän henkilöt sijaitsevat eri puolilla maailmaa, jonka vuoksi kehitys- ja dokumentaatiotyön tekeminen tapahtuu erilaisten sähköisten viestimien avulla. (6.)

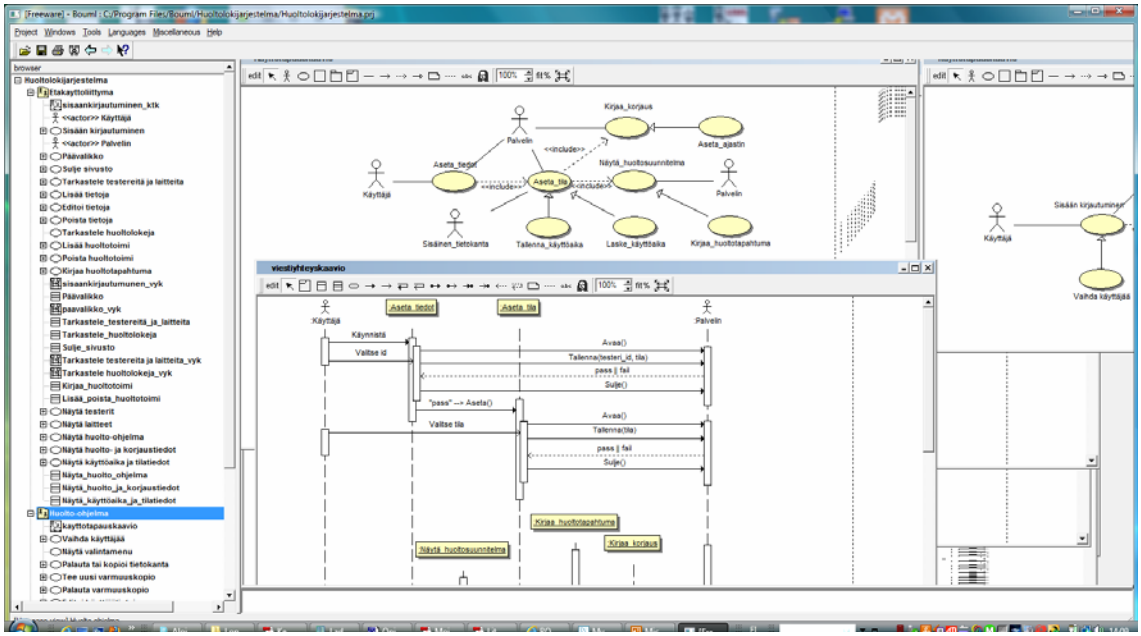
Apache sai alkunsa 1990-luvun puolivälissä. Ensimmäinen versio pohjautui National Center for Supercomputing Applications -yhtiölle kehitetyn palvelinohjelman pohjalta, johon oli eri Web-palveluiden tarjoajien toimesta tehty lukuisia pieniä lisäyksiä ja korjauksia yksittäisiin tarpeisiin. Myöhemmin pieni joukko näistä palveluntarjoajista aloitti keskinäisen yhteydenpidon sähköpostin välityksellä ja lopulta kokoontuivat yhteen sopimaan muutosten koordinoidusta tekemisestä. Yhteistyön tuloksena 8 jäseninen ydinryhmä perusti Apache Sofware Foundation, jonka tunnetuin tuote on Apache HTTP -palvelin. (6.)

3.3 Bouml

Bouml on ilmainen UML-kieltä (Unified Modeling Language) hyödyntävä ohjelma, jonka avulla voi luoda koodia erilaisille ohjelmointikielille, kuten C++, Java, PHP ja Pyhton (7). Bouml sisältää graafisten editorin, jonka avulla käyttäjä voi määritellä ohjelman rakenteen ja sisällön hyödyntämällä erilaisia suunnitteludiagrammeja (kuva 4). Ohjelman kaikki toiminnot mallinnetaan

diagrammeihin symbolien ja niille tehtävien parametrimäärittelyiden avulla. Ohjelmakoodi generoidaan valmiiden diagrammien pohjalta. Editoria voi hyödyntää ohjelman suunnittelemiseen ja dokumentaation tekemiseen myös ilman koodin generointia ja muillekin, kuin vain yllä mainituille ohjelmointikielille.

Boumlen on kehittänyt Bruno Pagès. Hän ilmoitti vuonna 2010 lopettavansa ohjelman jatkokehittämisen vedoten tekijänoikeusrikkomuksiin ja huonoon asioiden hoitamiseen Wikipedian hallinnon taholta.

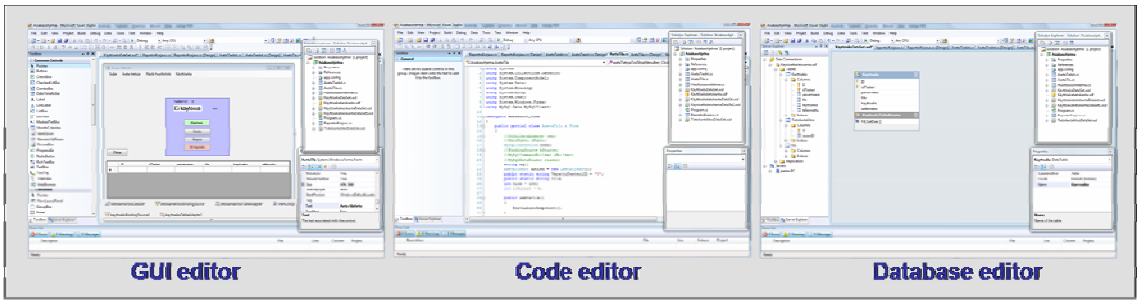


KUVA 4. Suunnitteludiagrammeja Bouml-editorin työpöydällä

3.4 MS Visual Studio

Microsoft-yhtiön julkaisema Visual Studio on yhdistetty kehitysympäristö (Integrated Development Environment), jonka avulla voidaan luoda konsoliohjelmia ja graafista käyttöliittymää (GUI) hyödyntäviä sovelluksia. Visual Studio on kaupallinen tuote, mutta se on opiskelijoille ilmainen. (8.)

Visual Studio tukee monia erilaisia ohjelmointikieliä, kuten C/C++, VB.NET, C# ja F#. Ohjelma sisältää sisäänrakennettuja työkaluja GUI-sovellusten, Web-sivujen ja tietokantojen suunnitteluun (kuva 5).



KUVA 5. Erilaisia suunnittelunäkymiä MS Visual Studio -editorissa

4 TAVOITEASETANTA

Alkuperäinen tavoite oli suunnitella ja toteuttaa tietokantapohjainen huoltolokijärjestelmä elektromekaniikkatuotannon testauslaitteille, joiden tulee täyttää ylläpidon osalta ISO9001-sertifikaatin sekä elinkaarenjälkeisen kierrätyksen osalta ISO14001-sertifikaatin asettamat vaatimukset. Koska työlle ei ollut varsinaista ulkoista tilaajaa tai pilottiasiakasta, oli sen puitteissa tarkoitus kartoittaa myös olemassa olevat kaupalliset vaihtoehdot, tunnistaa mahdollinen potentiaalinen asiakasryhmä ja miettiä, kuinka kehitettävä järjestelmä mahdollisesti tuottaisi lisäarvoa juuri heille ja olisi siten kilpailukykyinen vaihtoehto jo markkinoilta löytyville ratkaisuille.

Kaupallisten vaihtoehtoja kartoitettiin internetistä. Tutkimuksessa hyödynnettiin internethakupalveluita. Järjestelmien soveltuvuutta arvioitiin internetissä vapaasti saatavilla olevien sähköisien esittelymateriaalien perusteella. Tutkimustulosten pohjalta todettiin, että kaupallisia ratkaisuja on saatavilla useita. Niille kaikille on ominaista internetselaimessa toimiva käyttöliittymä ja laaja-alainen sisältö, jossa huoltolokityyppinen toiminta vaikutti olevan jokseenkin sivuroolissa. Mikään tutkituista järjestelmistä ei keskittynyt yksistään elektromekaanisten laitteiden tuotannontestauksessa käytettäviin mittalaitteisiin tai -laitteistoihin, vaan niiden käyttökohteet pyrkivät kattamaan mahdollisimman monien teollisuusalojen tarpeet omaisuudenhallinnan, laiteylläpidon ja tuotannonohjauksen alueilla.

Edellisen perusteella järjestelmän kehitystyölle asetettiin seuraavat yleiset tavoitteet:

- Kohderyhmänä ovat pienet kotimaiset elektromekaanisten tuotteiden valmistajat, joilla ei välttämättä ole omaa IT-tukihenkilöstöä.

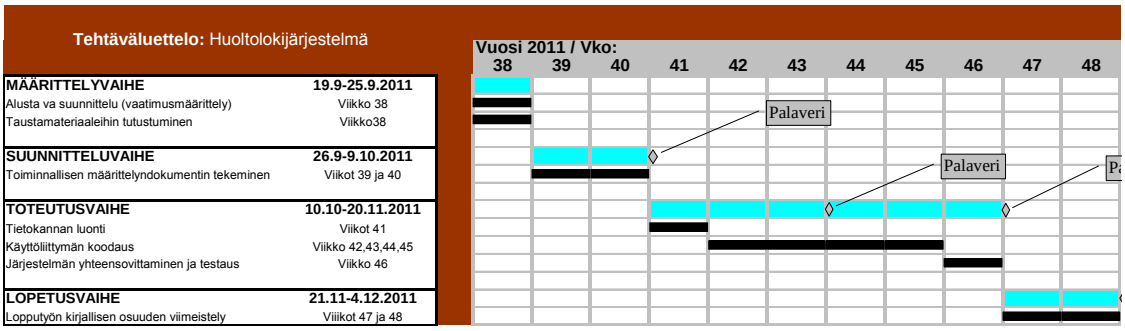
- Kehitetään järjestelmä, joka keskittyy pelkästään testauslaitteiden huolto-, korjaus- ja ylläpitotoimintaan.

- Tehdään suomenkielinen ja helppokäyttöinen ohjelmisto, joka ei vaadi merkittävissä määrin käyttäjäkoulutusta käyttöönoton yhteydessä tai ylläpidossa.

- Tehdään erilainen käyttöliittymä kaupallisiin ratkaisuihin verrattuna.

5 JÄRJESTELMÄN TOTEUTTAMINEN

Järjestelmä toteutettiin nelivaiheisena projektina, jonka alkuperäinen suunniteltu kesto oli 11 viikkoa (kuva 6). Projektin vaiheita olivat määrittely-, suunnittelu-, toteutus- ja lopetusvaihe.



KUVA 6. Toteutusprojektin tehtäväluettelo ja aikajanakaavio

5.1 Alustava määrittely

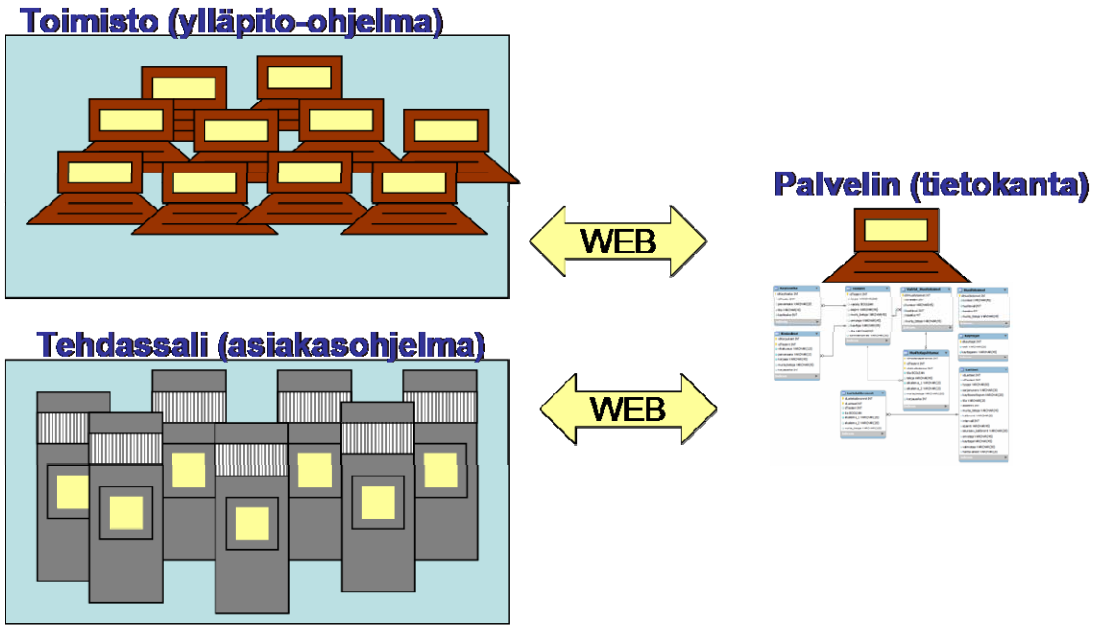
Ennen varsinaista suunnittelua tehtiin alustava määrittely yleisten vaatimusten ja tavoitteen pohjalta. Tässä vaiheessa mietittiin yksityiskohtaisesti, mitä toiminnallisuutta järjestelmän tuli sisältää, kuka käyttää sitä, miten ja missä tilanteissa ja minkälaisia tietoja tai syötteitä eri vaiheissa käsitellään. Kaikkia järjestelmän osia (tietokanta, asiakasohjelma, ylläpito-ohjelma ja etäkäyttöliittymä) käsiteltiin omina yksikköinään ja niiden toteutukseen valittiin sopiva alusta sekä ohjelmointikieli.

5.2 Huoltolokijärjestelmän yleinen kuvaus

Huoltolokijärjestelmän tarkoituksena on tuottaa palveluita, joiden avulla sitä käyttävä organisaatio voi hallinnoida käytössään olevia testilaitteistoja ja laitteita sekä ohjata niiden säännönmukaista ylläpito- ja korjaustoimintaa tehokkaasti ja vallalla olevien asetusten mukaisesti. Järjestelmän avulla organisaatio kykenee tarkastelemaan kohteille tehtyjä huoltoja, kalibrointeja ja korjauksia sekä tulossa olevia ennakkoon määritettyjä ylläpitotapahtumia. Laatuauditoinnissa organisaatio kykenee esittämään järjestelmään taltioitujen tietojen avulla riittävän näytön siitä, että laitteet ja laitteistot täyttävät ylläpidon osalta ISO9001-standardin asettamat vaatimukset. Lisäksi järjestelmän avulla

voidaan kerätä testilaitteistojen tila- ja käyttöaikatietoja sekä tietoa laitteiden elinkaarenjälkeisen kierrätyksen avuksi siitä, mitkä laitteet on valmistettu ISO14001-standardin vaatimusten mukaisesti.

Huoltolokijärjestelmä koostuu web-palvelimella sijaitsevasta tietokannasta, testilaitteistoihin asennettavasta asiakasohjelmasta sekä organisaation henkilöiden työasemiin asennettavasta ylläpito-ohjelmasta. Järjestelmäkomponenttien välinen tietoliikenne tapahtuu web-rajapinnan kautta (kuva 7).

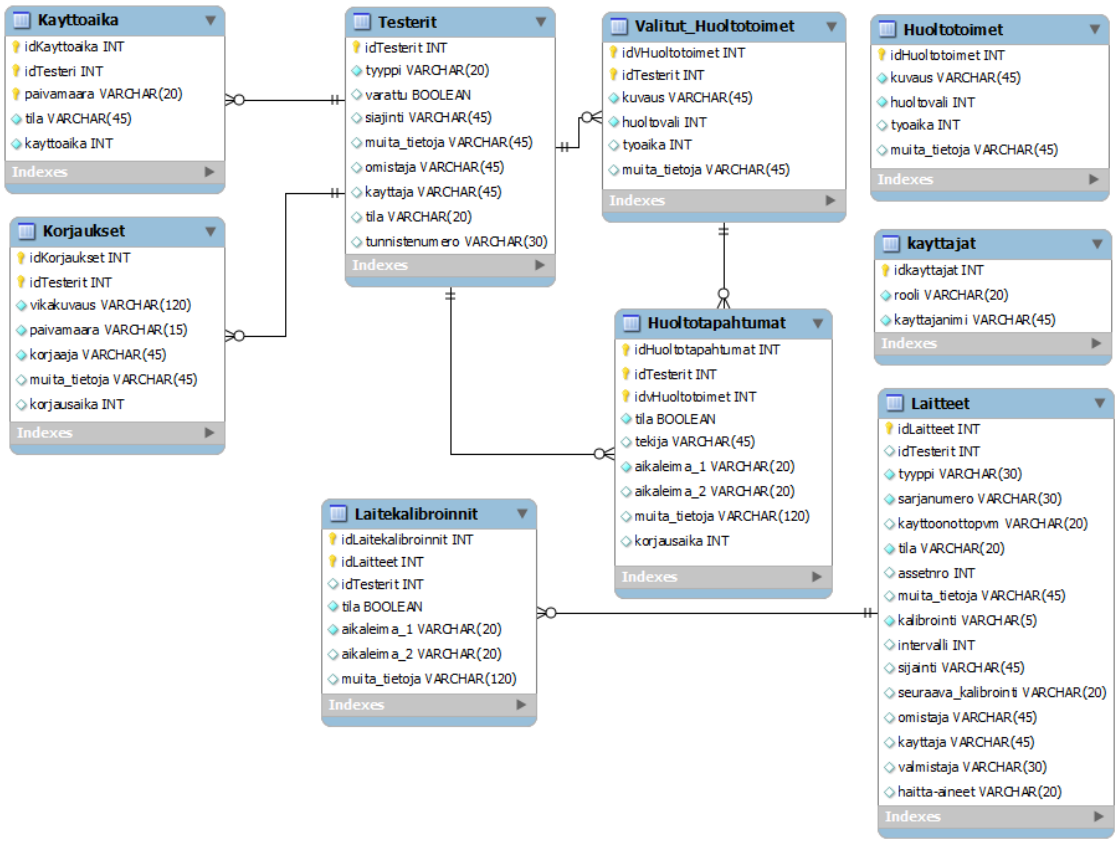


KUVA 7. Huoltolokijärjestelmän osien sijainti ja tietoliikenneyhteydet

5.3 Tietokanta

Tietokannan alustaksi valittiin helposti saatavilla oleva ja ilmainen MySql. Tietokannan suunnittelu aloitettiin miettimällä, millaisia tietoja järjestelmässä on tarve käsitellä ja tallentaa. Kun tiedot oli listattu, mietittiin niille sopiva tietotyyppi ja ne mallinnettiin tietokannan tauluihin (kuva 8). Relaatiotietokannalle ominaiseen tapaan tietokannassa säilytettävien tietojen määrä pyrittiin optimoimaan mahdollisimman pitkälle siten, että kutakin tietoa säilytettäisiin vain yhdessä paikassa. Tietojen tallentamisen tarkentamiseksi ja

nopeuttamiseksi tauluille määritettiin pääavaimet, sekä taulujen välille määritettiin yhteydet (viite-eheys). Taulujen välisten yhteyksien avulla varmistetaan myös se, että tietojen poistamisen ja päivityksen yhteydessä tietokannan yhtenäisyys säilyy.

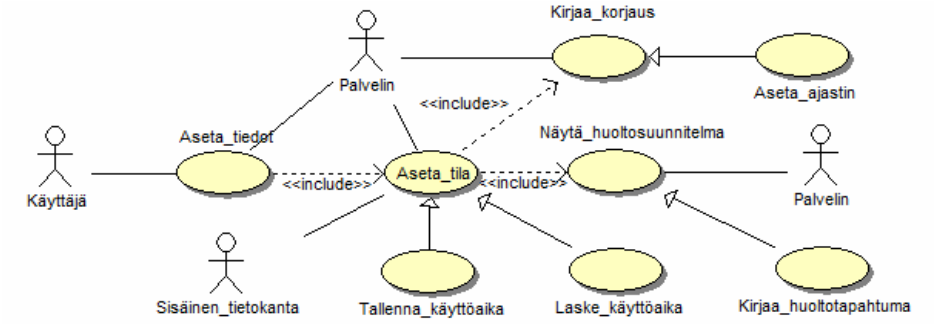


KUVA 8. Huoltolokitietokannan taulut ja tietotyypit (EER-kaavio). Taulun pääavaintietojen edessä on keltainen avainsymboli

Tietokannan suunnitteluun ja kannan luomiseen tarvittavan koodin generoimiseen käytettiin MySql Workbench -editoria (kts. 3.1). Liitteessä 1 on esimerkki työkalun avulla tehdystä MySql-skriptistä, joka luo huoltotoimet-nimisen taulun.

5.4 Asiakasohjelma

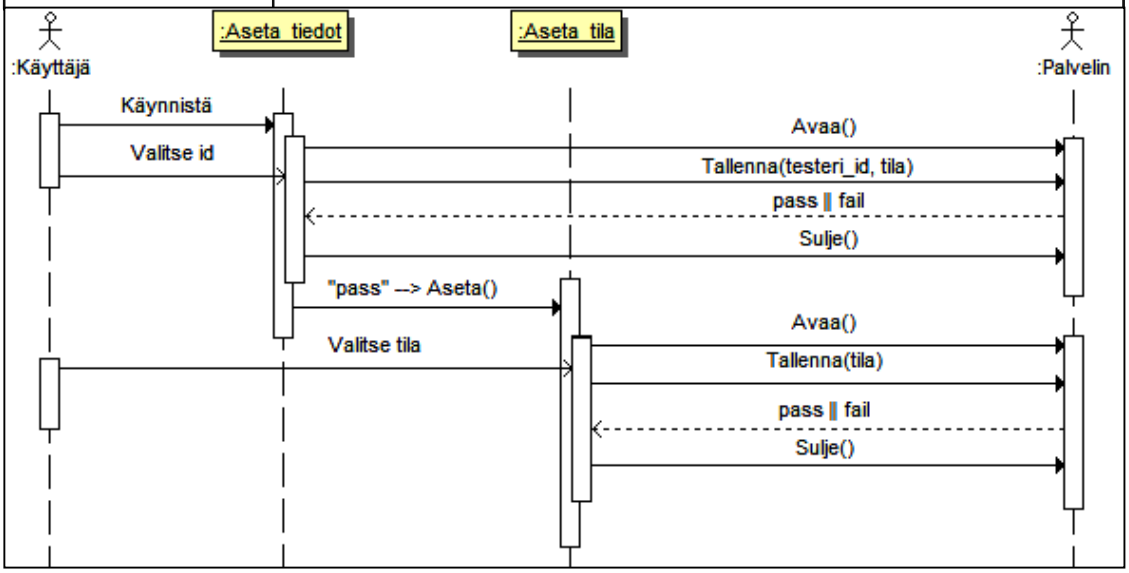
Asiakasohjelma suunniteltiin testilaitteistojen huoltajien käyttöön. Ohjelman suunnittelu aloitettiin tekemällä käyttötapauskaavio, johon hahmoteltiin karkeasti ohjelman sisältämä toiminnallisuus (kuva 9).



KUVA 9. Asiakasohjelman toiminnallisuus kuvattuna käyttötapauskaaviona

Kun vaadittava toiminnallisuus oli ylätasolla selvillä, kirjoitettiin kuvatut käyttötapaukset auki yksittäin ja niistä tehtiin viestiyhteyskaaviot ohjelmointityön pohjaksi (kuva 10).

Käyttötapaus 1 (Aseta tila)	Käyttäjä käynnistää asiakasohjelman ensimmäistä kertaa. Näytölle avautuu tietojen asetusikkuna. Näytöllä on valintaruutu, josta käyttäjä voi valita kyseisen testilaitteiston tunnisteen, jos se on vapaana. Klikkaamalla ”ok” ohjelma tallentaa valitun tunnisteen varauksen tietokantaa. Näytölle tulostuu teksti onnistuneesta testilaitetunnisteen ja asiakasohjelma -liitosparin muodostamisesta. Seuraavaksi näytölle avautuu tilan asetus formi, josta käyttäjä pääsee valitsemaan laitteiston tilan (käytössä, huolto, korjaus tai ei käytöstä). Tilan valinta suoritetaan klikkaamalla haluttua tilapainiketta, jolloin tilamuutos tallentuu tietokantaan. Virhetilanteet: -puutteelliset tai väärät syötetiedot -ei yhteyttä palvelimeen -testilaitetunniste ei ole vapaana

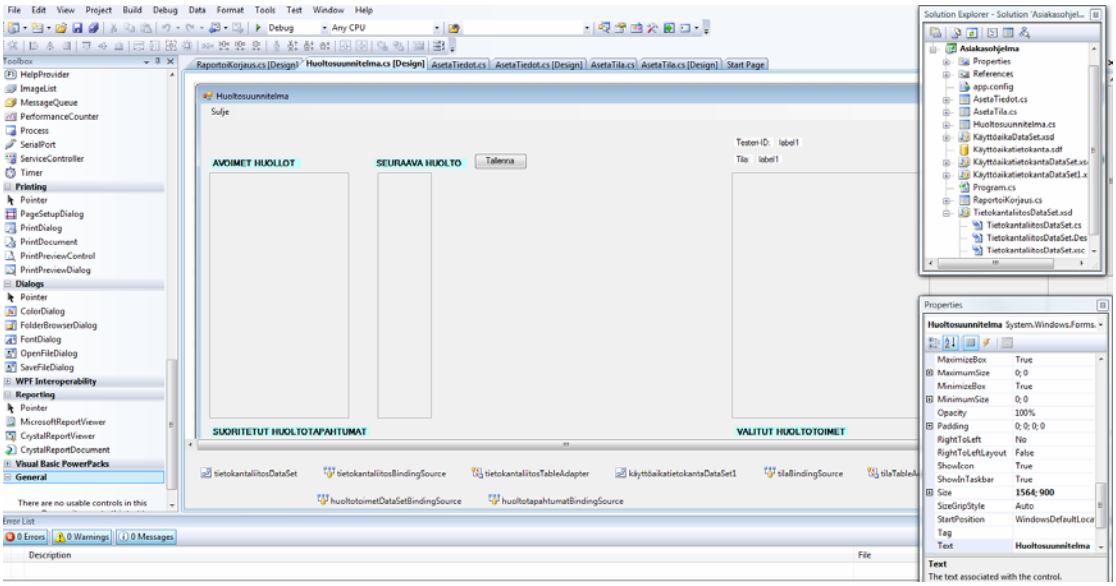


KUVA 10. Yksittäisen käyttötapauksen sanallinen kuvaus ja viestiyhteyskaavio

Asiakasohjelman käyttöympäristönä on testilaitteiston ohjaintietokone. Erillisellä testilaitteiston ohjaimessa suoritettavalla ohjelmalla tavoitellaan helppokäyttöisyyttä. Suunnittelun lähtökohtana oli se, että huolto- tai korjaustoimenpiteen suorittanut henkilö voi kirjata tehdyn työn etätietokantaan mahdollisimman vaivattomasti ja nopeasti suoraan kyseiseltä kohteelta ilman erillistä kirjautumista palvelimelle. Tämä poikkeaa merkittävästi kaupallisten ratkaisujen yleisestä tavasta, jossa käyttäjää vaaditaan kirjautumaan järjestelmään, ennen kuin tietojen syöttäminen on mahdollista.

Toinen merkittävä helppokäyttöisyyttä tukeva toiminto, joka asiakasohjelmaan suunniteltiin, on se, että ohjelma linkitetään tietokantaan juuri sen testilaitteiston tietoihin, johon se on asennettu. Tämä tuo merkittävää etua etenkin silloin, jos testilaitteistoja on paljon, useita kymmeniä, tai jopa satoja. Käyttäjän ei tarvitse etsiä tietyn testilaitteiston tietoja yhteisestä kannasta, vaan linkityksen jälkeen ohjelma näyttää oikean laitteiston tiedot automaattisesti. Asiakasohjelman linkitys (tietokantaliitos) tarvitsee suorittaa vain yhden kerran ensimmäisen käynnistyksen yhteydessä. Ohjelma muistaa liitoksen myös tietokoneen uudelleenkäynnistämisen jälkeen.

Asiakasohjelman koodaus ja graafinen käyttöliittymäsuunnittelu tehtiin MS Visual Studio -ohjelmointialustalla C#-ohjelmointikielellä (kuva 11).



KUVA 11. Huoltolokinäkymä (C# Windows Forms Application) suunnitteluеditorin näytöllä

Käyttöliittymäliittymäsuunnittelussa huomioitiin helppokäyttöisyyden lisäksi syötevirheiden minimointi. Syötevirheitä pyritään minimoimaan lähinnä seuraavilla neljällä tavalla

- tarkistamalla aina syötteiden lähetyksen yhteydessä se, että käyttäjä on lisännyt tietoja kaikkiin tarvittaviin kenttiin

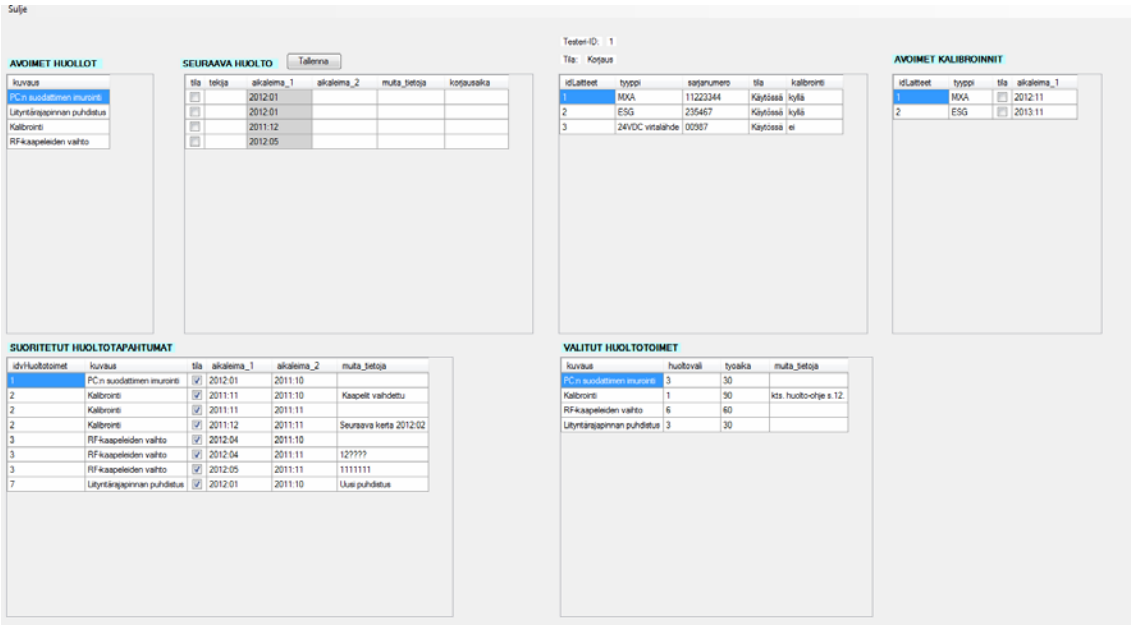
- hakemalla kannasta oikeat arvot valintalaatikoihin käyttäjän valittavaksi silloin, kun valinta tulee tehdä joltain tietyltä rajatulta alueelta, esimerkiksi laitetunnus 1−5

- näyttämällä tietyt valinnat ja painikkeet vasta siinä vaiheessa, kun tarvittavat esivaiheet on suoritettu

- asettamalla taulun pääavaimen automaattisesti tietojensyöttölomakkeeseen.

Huoltotoimien luominen suoritetaan aina ennen huoltolokinäkymän luomista. Ohjelma tarkistaa siihen linkitetylle testilaitetunnukselle löytyvät huoltotoimet tietokannasta käyttäjän asettamaa testilaitekohtaista huolto-ohjelmaa vasten.

Esimerkiksi, jos testilaitteen huolto-ohjelma sisältää suodattimen imuroinnin, joka on määrä tehdä 3 kuukauden välein, tarkistaa ohjelma tietokannasta kyseiseen huoltotehtävään liittyvät tapahtumat. Jos yhtään avointa tapahtumaa ei löydy, luodaan avoin huoltotoimi, jonka määräaika asetetaan 3 kuukauden päähän tarkasteluajankohdasta. Jos löytyy avoin odottava huoltotoimi, ei tehdä mitään. Jos löytyy huoltotoimi, joka on kuitattu tehdyksi, luodaan uusi odottava huoltotoimi 3 kuukauden päähän tarkasteluajankohdasta. Kun huoltotoimien päivitys on suoritettu, tulostetaan dynaaminen huoltolokinäkymä näytölle (kuva 12).



KUVA 12. Huoltolokinäkymä ohjelmaa suoritettaessa

Käyttöaikatiedon kerääminen ei alun perin kuulunut asiakasohjelman suunniteltuun toiminnallisuuteen, mutta se päätettiin toteuttaa, koska sen katsottiin tuovan lisäarvoa valituille ratkaisuille, jotka tehtiin ohjelman toteutuksen suhteen (testilaiteohjaimessa palveluna suoritettava ohjelma). Käyttöajan keräämistä suoritetaan aina, kun ohjelman suoritus on käynnissä. Erilaisia tiloja on 4 (käytössä, huolto, korjaus ja ei käytössä). Oletustila ensimmäisen käynnistyksen ja tietokantaliitoksen luomisen jälkeen on aina ”ei käytössä”.

Asiakasohjelman perusnäyttö on tilan asetuslomake, josta käyttäjä voi asettaa käyttötapaa vastaavan tilan. Ohjelma tarkistaa 10 sekunnin välein vallitsevan tilan ja kasvattaa kyseisen tilan laskuria vastaavasti. Jotta käyttöajan katkoton

kerääminen voidaan taata myös mahdollisten yhteyskatkosten aikana, suoritetaan laskuritietojen kerääminen ensimmäisessä vaiheessa tilatiedon tavoin sisäiseen tietokantaan. Käyttöaikatietoa kerätään vuorokauden ajan kerrallaan kunkin tilan omaan laskuriin ja vuorokauden vaihduttua kerätyt laskurinarvot siirretään palvelimella sijaitsevaan tietokantaan, ja siirretyt tiedot siivotaan pois sisäisestä tietokannasta. Alla on koodiesimerkkejä käyttöaikalaskureiden luomisesta, laskurinarvon kasvattamisesta, käyttöaikatietojen siirtämisestä etätietokantaan ja vanhojen tietojen poistamisesta.

```
if (teeLaskurit == true)
{ // luodaan laskurit
    kayttoaikaTableAdapter1.Insert(System.Convert.ToInt32(labelTesteriID.Text),
        nykyinenPäivämäärä, "Ei käytössä", 0, "false");
    kayttoaikaTableAdapter1.Insert(System.Convert.ToInt32(labelTesteriID.Text),
        nykyinenPäivämäärä, "Huolto", 0, "false");
    kayttoaikaTableAdapter1.Insert(System.Convert.ToInt32(labelTesteriID.Text),
        nykyinenPäivämäärä, "Korjaus", 0, "false");
    kayttoaikaTableAdapter1.Insert(System.Convert.ToInt32(labelTesteriID.Text),
        nykyinenPäivämäärä, "Käytössä", 0, "false");
    päivitäDataGrid();
}
```

Käyttöaikalaskureiden luominen suoritetaan tauluadapterin avulla. Tauluadapteri, tietokannassa oleva käyttöaikataulu ja näytölle tulostettava tietoruudukko ovat sidottu toisiinsa siten, että niissä kaikissa on aina sama tieto.

```
while (i < rivejä - 1)
{
    if (dataGridViewKäyttöaika.Rows[i].Cells[1].Value.ToString() == VarattuTesteriID &&
        dataGridViewKäyttöaika.Rows[i].Cells[2].Value.ToString() == nykyinenPäivämäärä &&
        dataGridViewKäyttöaika.Rows[i].Cells[3].Value.ToString() == tila &&
        dataGridViewKäyttöaika.Rows[i].Cells[5].Value.ToString() == "false")
    {
        laskurinArvo = System.Convert.ToInt32(dataGridViewKäyttöaika.Rows[i].Cells[4].Value.ToString());
        dataGridViewKäyttöaika.Rows[i].Cells[4].Value = laskurinArvo + 10; // +10sek
        labelLaskuri.Text = dataGridViewKäyttöaika.Rows[i].Cells[4].Value.ToString();
        päivitäDataGrid();
    }
    i++;
}
private void päivitäDataGrid()
{
    this.Validate();
    kayttoaikaBindingSource1.EndEdit();
    kayttoaikaTableAdapter1.Update(käyttöaikaDataSet.Kayttoaika);
    this.kayttoaikaTableAdapter1.Fill(this.käyttöaikaDataSet.Kayttoaika);
}
```

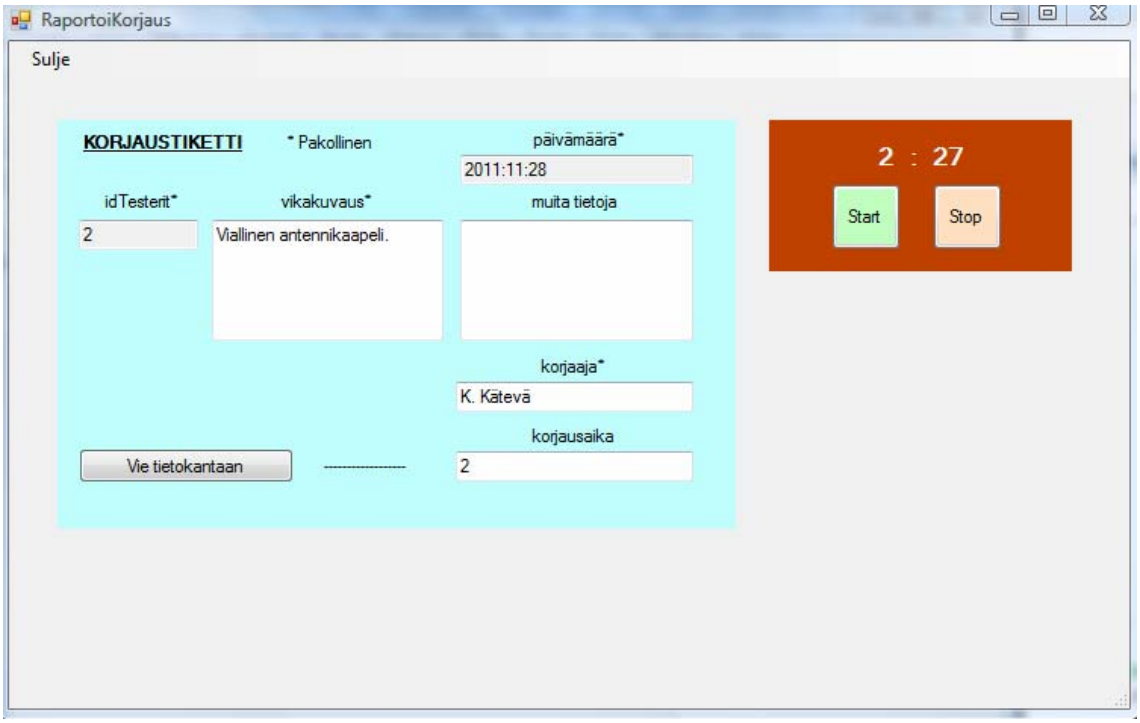
Vallitsevan tilan laskurinarvon kasvattaminen suoritetaan suoraan näytöllä olevaan tietoruudukkoon, josta tieto päivittyy tietokantaan, kun päivitäDataGrid-metodi on suoritettu.

```
while (i < dataGridViewKäyttöaika.Rows.Count - 1)
{

    if (DateTime.ParseExact(dataGridViewKäyttöaika.Rows[i].Cells[2].Value.ToString(), "yyyy:MM:dd", null) <
        DateTime.ParseExact(nykyinenPäivämäärä, "yyyy:MM:dd", null) &&
        dataGridViewKäyttöaika.Rows[i].Cells[5].Value.ToString() == "false")
    {
        sql = @"INSERT INTO kayttoaika (idTesteri,paivamaara,tila,kayttoaika) VALUES(" +
            dataGridViewKäyttöaika.Rows[i].Cells[1].Value.ToString() + ",'" +
            dataGridViewKäyttöaika.Rows[i].Cells[2].Value.ToString() + "','" +
            dataGridViewKäyttöaika.Rows[i].Cells[3].Value.ToString() + "'," +
            dataGridViewKäyttöaika.Rows[i].Cells[4].Value.ToString() + ")";
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        cmd.ExecuteNonQuery();
        dataGridViewKäyttöaika.Rows[i].Cells[5].Value = true; // Vaihdetaan laskurintiedon tila gridiin = tallennettu
        dataGridistäPoistettavatRivit[j] = i; j++;
        conn.Close();
    }
    i++;
}
if (j > 1)
{ // poistetaan onnistuneesti siirretyt rivit gridistä
    int k = j - 1; // jos poistettavia rivejä on esim 8, j = 9
    while (k > 0)
    {
        try
        {
            dataGridViewKäyttöaika.Rows.RemoveAt(dataGridistäPoistettavatRivit[k]); k--;
        }
        catch { MessageBox.Show("Laskurin poistaminen dataDridistä ei onnistunut!"); }
    }
    päivitäDataGrid();
}
```

Ohjelma tutkii aina käyttöaikalaskureiden päiväyksen laskuriarvon päivityksen yhteydessä. Jos aikaleima on edelliseltä päivältä tai vanhempi, siirretään vanhat laskurinarvot etätietokantaan ja poistetaan onnistuneesti siirretyt tiedot sisäisestä tietokannasta.

Korjaustapahtumien raportointitarkoitukseen tehtiin oma käyttöliittymänäkymä, joka koostuu tietojensyöttölomakkeesta ja ajastimesta (kuva 13). Näkymä avautuu näytölle, kun käyttäjä on asettanut testilaitteiston tilaksi ”korjaus”. Ajastintoiminnallisuus lisättiin käyttöliittymään helpottamaan korjaustiedon raportointia. Ajastin käynnistyy automaattisesti tilatiedon asettamisen yhteydessä ja tulostaa korjaukseen kulunutta aikaa näytölle. Korjaukseen kuluneen aikatiedon siirtäminen korjaustiketille tapahtuu tuplaklikkaamalla ajastimen aikanäyttöä.



KUVA 13. Korjaustapahtuman raportointinäkymä. Syöttövirheiden estämiseksi testauslaitteistotunnus ja päivämäärä asetetaan lomakkeeseen ohjelmallisesti

Asiakasohjelman looginen toiminta on kuvattu kokonaisuudessaan liitteessä 2 olevassa yksinkertaistetussa lohkokaaviossa.

5.5 Ylläpito-ohjelma

Ylläpito-ohjelman suunnittelu ja toteutus eteni noudattaen samoja vaiheita, kuin yllä kuvattu asiakasohjelman suunnittelu ja toteutus (kappale 5.4). Ohjelman sisältämä toiminnallisuus hahmoteltiin ensin käyttötapauskaavioon, jonka pohjalta sitten yksittäiset käyttötapaukset kirjoitettiin auki ja niistä tehtiin viestiyhteyskaavio.

Ylläpito-ohjelman käyttöympäristö on ohjelmaa käyttävän organisaation työntekijöiden henkilökohtaiset tietokoneet. Ohjelman käyttäjiä ovat huoltotyön esimiehet, laitekalibrointityön suorittajat, testilaitteiden rakentamisesta, logistiikasta ja toteutuksesta vastaavat henkilöt, sekä muut organisaatiossa työskentelevät henkilöt, joiden tehtäviensä hoitamisessa tarvitsevat järjestelmän tarjoamia tietoja ja palveluita.

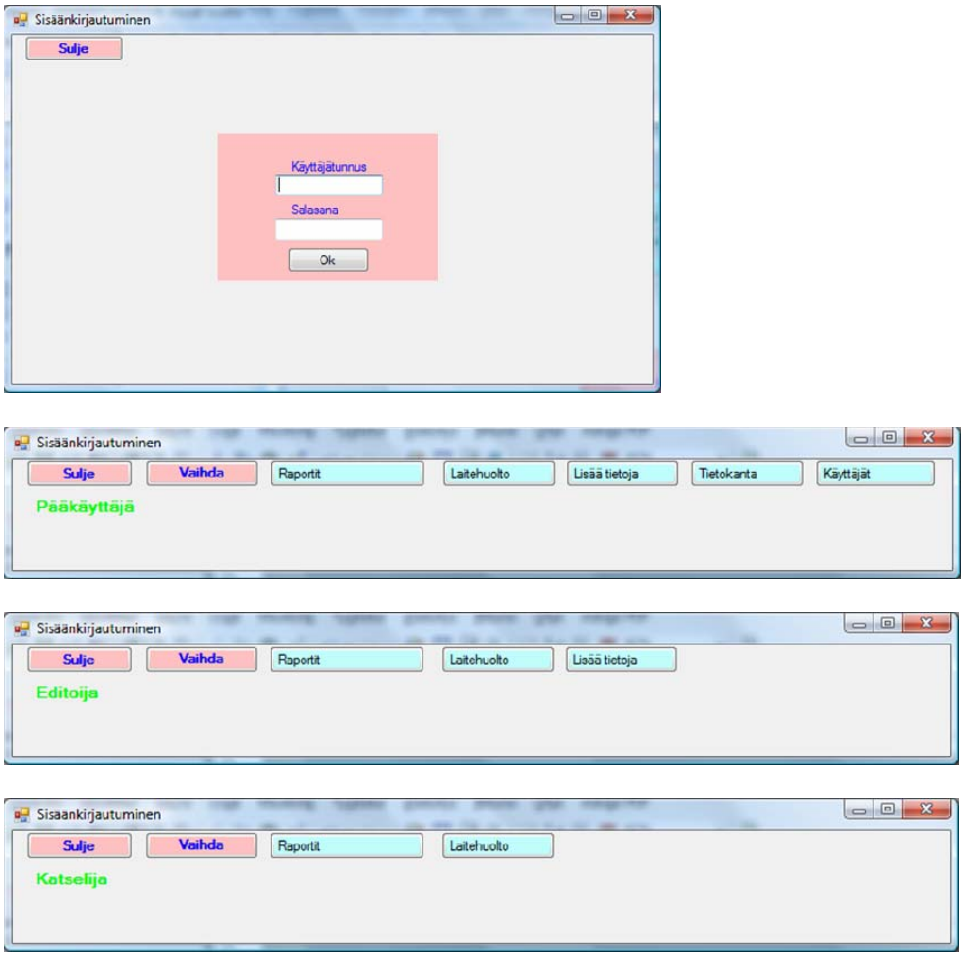
Koska ohjelma sisältää toiminnallisuutta, joka väärin käytettynä voi johtaa tietojen menetykseen tai vääristymiseen, lisättiin siihen kolme erilaista käyttäjäprofiilia seuraavin kuvaksin

- editoija, jolla on oikeus tarkastella tai muokata kaikkia järjestelmässä olevia tietoja, pois lukien käyttäjätiedot sekä toiminnallisuus liittyen tietokannan varmuuskopiointiin tai palautukseen

- katselija, jolla on ainoastaan oikeus tarkastella tietoja, jotka liittyvät laitteisiin, laitteistoihin, huolto- ja korjaustöihin sekä kerättyyn käyttöaikaan

- pääkäyttäjä, jolla on oikeus tarkastella tai muokata kaikkia järjestelmässä olevia tietoja, mukaan lukien käyttäjätiedot ja toiminnallisuus liittyen tietokannan varmuuskopiointiin ja palautukseen.

Ohjelma tunnistaa käyttäjäprofiilin sisään kirjautumisen yhteydessä, jolloin käyttäjää pyydetään antamaan käyttäjätunnus ja salasana. Kun sisään kirjautuminen on suoritettu onnistuneesti, tulostuu näytölle päävalikko, jossa on näkyvillä vain käyttäjäprofiilin mukaiset sallitut toiminnot (kuva 14).



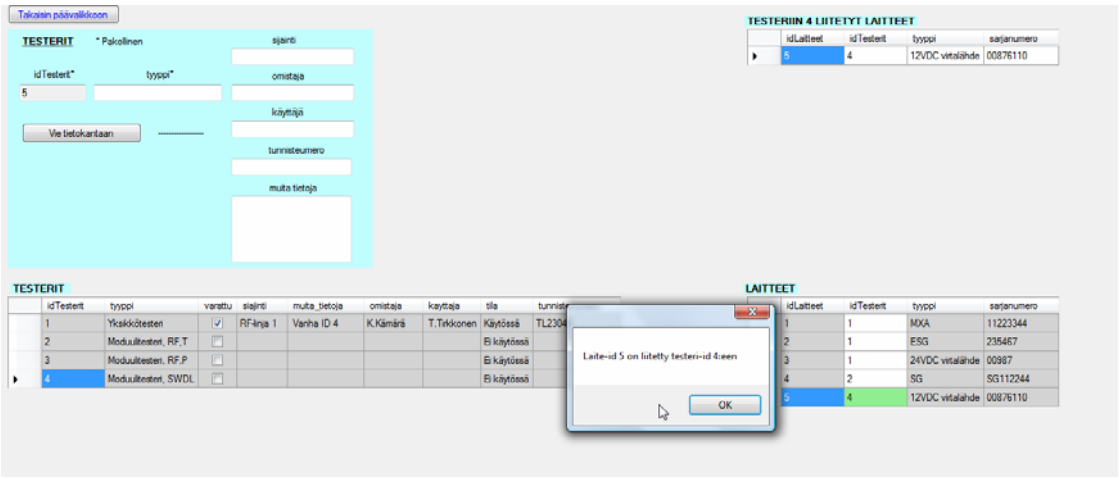
KUVA 14. Ylläpito-ohjelman sisään kirjautumis- ja valikkonäkymät eri käyttäjäprofiileille

Ylläpito-ohjelman käyttöliittymien suunnittelussa ja toteutuksessa pyrittiin helppokäyttöisyyteen ja syötevirheiden minimointiin samalla tavalla kuin asiakasohjelman kohdalla (luku 5.4). Ohjelman koodaus ja graafinen käyttöliittymäsuunnittelu tehtiin myös asiakasohjelman tavoin MS Visual Studio -ohjelmointialustalla C#-ohjelmointikielellä (C# Windows Forms Application). Ohjelman keskeisiä toiminnallisuuksia ovat laitetietojen syöttäminen järjestelmään, testilaitteistojen ja huolto-ohjelmien luominen, käyttäjätietojen hallinnointi, tietokannan ylläpito, testilaitteistojen käyttöajan ja tilatietojen näyttäminen, kalibrointitapahtumien luonti ja näyttäminen, huoltotapahtumien näyttäminen, sekä kierrätyslaitteiden hallinnointi.

Uusien tietojen (laite, laitteisto, tai huoltotoimi) lisääminen järjestelmään tehdään käyttötarkoitukseen räätälöityjen syöttölomakkeiden avulla. Syöttövirheiden minimoimiseksi lomakkeiden toteutuksessa on noudatettu

kappaleessa 5.4 esitettyjä 4:ää periaatetta. Lomakkeet on pyritty rakentamaan mahdollisimman selkeiksi ja helppokäyttöisiksi. Pakolliset syötekentät on merkitty tähdillä ja näytölle tulostetaan reaaliaikaisesti sen taulun tiedot tietokannasta, johon tietojen lisääminen tapahtuu. Lisäksi tallennustapahtuman onnistumisesta tai virhetilanteista tulostetaan näytölle viesti.

Yksittäisten laitteiden liittämisen johonkin tiettyyn testilaitteistoon voi tehdä kahdella eri tavalla. Laitteen voi liittää suoraan testilaitteistoon, joko laitetietojen syöttövaiheessa valitsemalla sopiva testilaitetunnus vastaavaan syöteruutuun, tai testilaitesyöttölomakkeessa hiiren avulla valitsemalla ensin testilaitetunnus ja sitten liitettävän laitteen tunnus (kuva 15). Jos laitteet ja laitteistot on syötetty valmiiksi tietokantaan, jälkimmäinen vaihtoehto tarjoaa visuaalisemman ja nopeamman tavan testilaitteiston kokoonpanon luomiseen ja modifioimiseen. Lisäksi käyttäjä näkee reaaliajassa editoinnin alla olevan testilaitteiston kokoonpanon sisällön.



KUVA 15. Laitetunnus 5 on liitetty testilaitteistotunnus 5:een graafisen käyttöliittymän avulla

Testilaitteiston huoltosuunnitelma koostuu yksittäisistä huoltotoimista, jotka on liitetty testilaitteistoon. Käyttäjä määrittelee yksittäiset huoltotoimet tietokantaan, josta niitä voi myöhemmin hakea ja liittää testilaitteistoihin. Saman huoltotoimen voi liittää useaan testilaitteistoon. Huoltotoimi yksilöityy vasta sen jälkeen, kun liitos on suoritettu. Esimerkiksi, jos käyttäjä on luonut tietokantaan PC:n imurointi -nimisen huoltotoimen ja hän liittää sen johonkin testilaitteistoon, muodostuu valittu huoltotoimi, jota järjestelmä käsittelee tästä eteenpäin

yksittäisenä tietueena, jolla on yksilölliset parametrit. Edellä kuvatun huoltotoimen monistamisen voi suorittaa periaatteessa rajattomia kertoja.

Ylläpito-ohjelman avulla voi luoda järjestelmään uusia käyttäjiä (katselija tai editoija), poistaa olemassa olevia käyttäjiä tai vaihtaa aktiivisien käyttäjätunnuksien salasanoja (kuva 16).

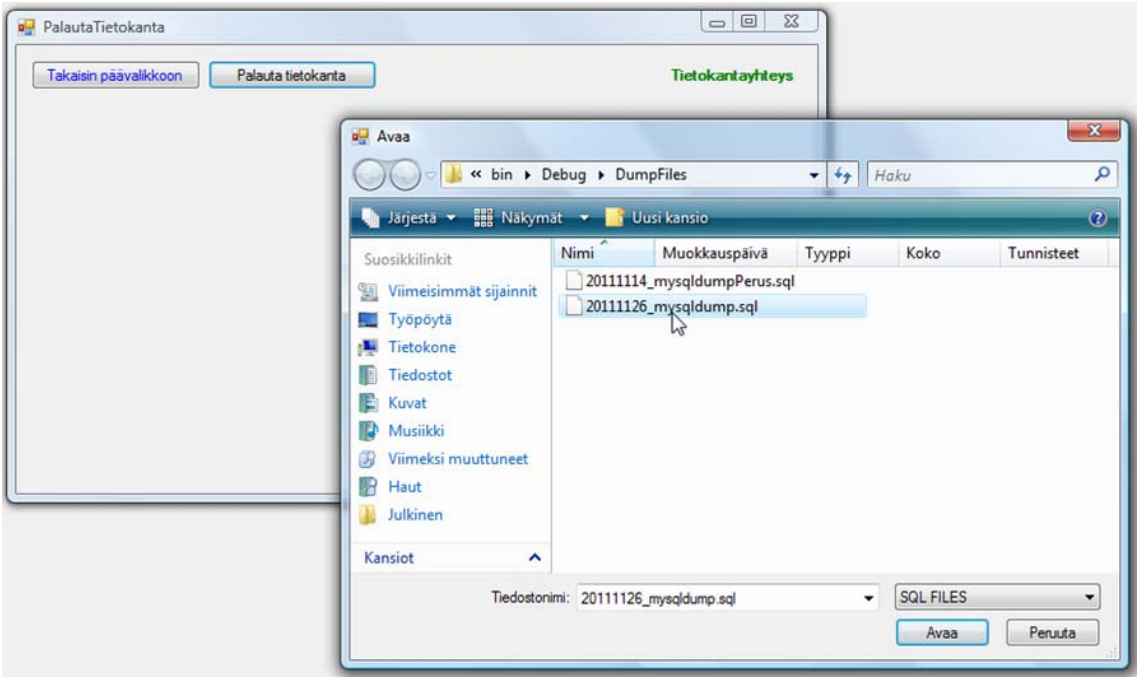
KUVA 16. Suorituksen aikaiset näytöt uuden käyttäjätunnuksen lisäämisestä, salasanan vaihtamisesta ja käyttäjätunnuksen poistamisesta

Käyttäjätunnukset ja salasanat tallentuvat MySql-tietokantaan. Järjestelmän pääkäyttäjän tunnus luodaan php-admin-työkalun avulla. Pääkäyttäjällä on kaikki oikeudet tietokantaan, muilla käyttäjillä käyttöoikeudet ovat roolin mukaisesti rajatut. Alla koodiesimerkki MySql-scripteistä, joiden avulla uudet käyttäjät luodaan.

```
if (radioButton_editor.Checked == true)
{
    sql = @"GRANT SELECT,INSERT,UPDATE ON huoltolokitietokanta.* TO '" + user +
        "'@'localhost' IDENTIFIED BY '" + passw + "'";
    rooli = "editor";
    id_käyttäjät = haeID();
}
else if (radioButton_viewer.Checked == true)
{
    sql = @"GRANT SELECT ON huoltolokitietokanta.* TO '" + user +
        "'@'localhost' IDENTIFIED BY '" + passw + "'";
    rooli = "viewer";
    id_käyttäjät = haeID();
}
```

Tietokannan ylläpitotoimintojen suunnittelussa ja toteutuksessa huomioitiin se, että järjestelmää käyttävän organisaation henkilöstöllä ei välttämättä ole tarvittavia tietoja ja taitoja sellaisten toimenpiteiden suorittamiseen, jotka tapahtuvat osin, tai kokonaan järjestelmän ulkopuolella ja vaativat järjestelmän ulkopuolisten työkalujen käyttöä. Toiminnoista haluttiin niin yksinkertaisia, että niitä voi käyttää ilman ulkopuolista apua tai ohjedokumentaatiota. Lisäksi tuli huomioida ja turvata tietojen säilyvyys, sillä osaamattoman käytön vuoksi tietoja voitaisiin menettää merkittävästi.

Edellä kuvattujen vaatimusten pohjalta toteutettiin käyttöliittymänäyttö ja toiminnallisuus, joka mahdollistaa tietokannan varmuuskopioinnin, tai palautuksen napin painalluksella (kuva 17). Ohjelmassa hyödynnetään MySql-palvelimen mukana tulevia ohjelmia mysqldump.exe ja mysql.exe, sekä C#-ohjelmointikielen luokkia StreamWriter, StreamReader ja ProcessStartInfo. Mysqldump.exe ohjelmaa käytetään tietokannan varmuuskopiointiin. Se luo tietokannasta komentojonosarjan, joka voidaan tallentaa valittuun kohteeseen erillisenä tiedostona. Mysql.exe ohjelmaa käytetään tietokannan palauttamiseen. Sen avulla voidaan palauttaa tietokanta edellä kuvatun kaltaisen komentojonosarjan pohjalta. StreamWriter-luokan palveluita hyödynnetään tietokannan varmuuskopioinnin yhteydessä kirjoittamaan merkkijonomuuttujan sisällöksi luettu komentojonosarja sql-tiedoston sisällöksi, sekä tietokannan palautuksen yhteydessä kirjoittamaan sql-tiedostosta luettu komentosarja mysql.exe ohjelman syötteeksi. StreamReader-luokan palveluita käytetään palautuksen yhteydessä lukemaan sql-tiedostoon tallennettu komentojonosarja.



KUVA 17. Ohjelma on avannut tiedostoselaimen näytölle, josta käyttäjä voi valita palautettavan varmuuskopiotallenteen

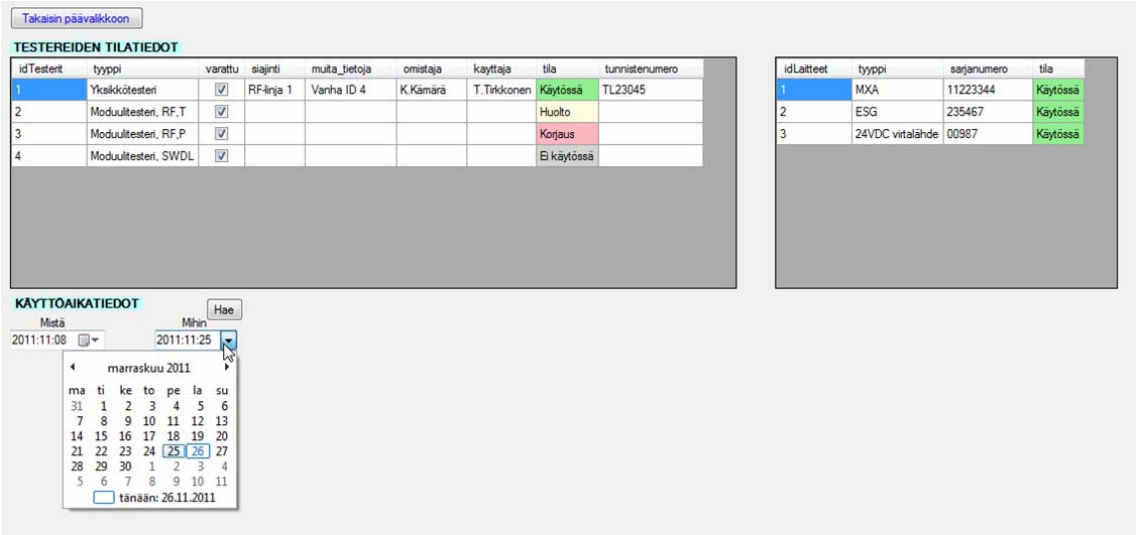
Tietokannan varmuuskopiointi suoritetaan siten, että ohjelma käynnistää mysqldump.exe-ohjelman, joka muodostaa kopioitavan tietokannan sisällöstä

komentojonosarjan ja tulostaa sen standardiulostuloon, josta ohjelma edelleen lukee sen merkkijonomuuttujan sisällöksi. Kun tuloste on valmis, kirjoittaa ohjelma sen perusteella varmuuskopiotiedoston ennalta määrättyyn tiedostokansioon. Tiedoston päätteeksi tulee ”.sql” ja tunnisteeksi nimen lisäksi senhetkinen päivämäärä. Alla on koodiesimerkki varmuuskopiotallenteen tekemisestä.

```
StreamWriter file = new StreamWriter(path);
ProcessStartInfo psi = new ProcessStartInfo();
psi.FileName = @"C:\xampp\mysql\bin\mysqldump";
psi.RedirectStandardInput = false;
psi.RedirectStandardOutput = true;
psi.Arguments = string.Format(@"-u{0} -p{1} -h{2} {3}", un, pw, "localhost", "huoltolokitietokanta");
psi.UseShellExecute = false;
Process process = Process.Start(psi);
string output;
output = process.StandardOutput.ReadToEnd();
file.WriteLine(output);
process.WaitForExit();
file.Close();
process.Close();
```

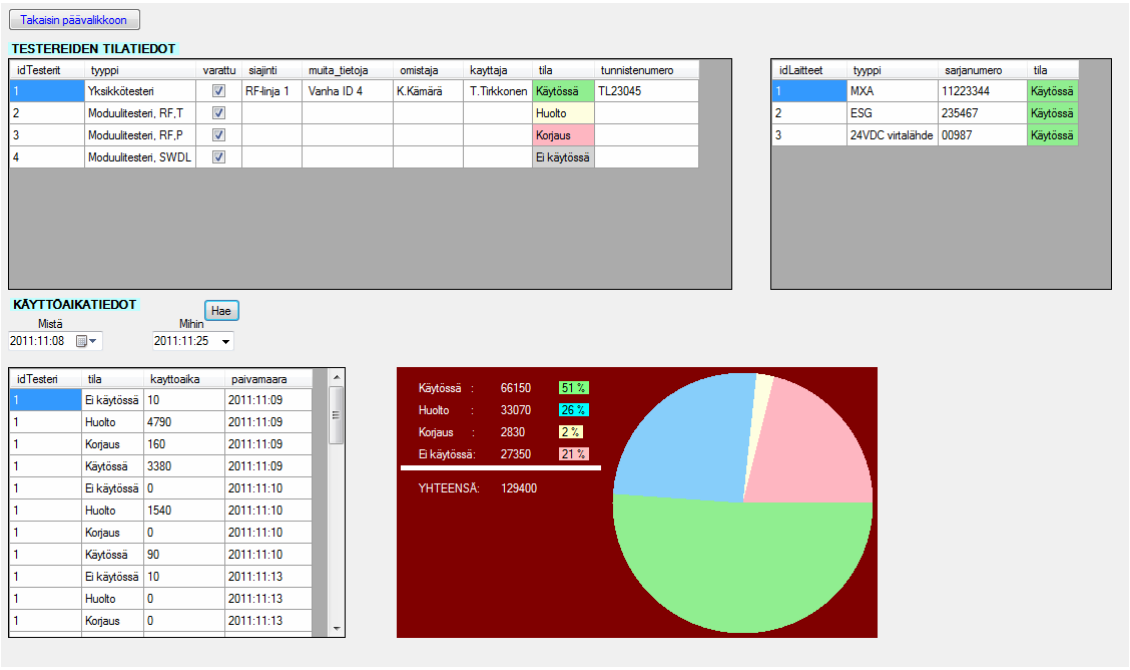
Tietokannan palauttaminen tapahtuu periaatteessa suorittamalla edellä kuvattu prosessi käänteisessä järjestyksessä. Ennen palautusprosessin käynnistämistä ohjelma tulostaa näytölle tiedostoselaimen ja pyytää käyttäjää valitsemaan palautettavan varmuuskopiotallenteen. Kun varmuuskopiotallenne on valittu, käynnistää ohjelma mysql.exe-ohjelman ja lukee standardi sisääntulon avulla sen syötteeksi varmuuskopiotallenteen sisältämän komentojonosarjan, jonka perusteella mysql.exe palauttaa tietokannan.

Testilaitteiston tila- ja käyttöaikatietojen esittämiseen tarvittavan toiminnallisuuden suunnittelun ja toteutuksen tavoitteena oli helppokäyttöisyys ja visuaalisuus. Vaatimusten pohjalta toteutettiin käyttöliittymänäyttö, jonka avulla testilaitteistojen tila- ja käyttöaikatietoja voi selata vaivattomasti pelkkien hiirellä tehtävien ohjauskomentojen avulla. Ohjelma tulostaa aloitusnäyttöön kaikki tietokannassa olevat testilaitteet ja niiden senhetkiset tilatiedot. Tilatietojen visualisointi näytöllä tapahtuu värien avulla. Kun käyttäjä klikkaa hiirellä jonkin testilaitteen tunnistetta, tulostuu näytölle edellisten lisäksi valittuun mittalaitteistotunnisteeseen liitetyt laitteet ja niiden tilat, sekä käyttöaikatietojen tarkasteluajanjakson aloitus- ja lopetuspäivämäärien asettamiseen käytettävät valitsimet (kuva 18).



KUVA 18. Käyttöaikatietojen aloitus- ja lopetuspäivämäärien asettaminen valitulle kohteelle tapahtuu kalenterivalitsimien avulla

Kun käyttöaikatietojen aloitus- ja lopetuspäivämäärät on asetettu, hakee ohjelma valitun kohteen käyttöaikatiedot asetetulla ajanjaksolla ja tulostaa hakutiedot näytölle ja piirtää niistä piirakkadiagrammin (kuva 19).



KUVA 19. Kuvassa on valitun kohteen käyttöaika- ja tilatietoja tarkasteltavana

Testilaitteistot koostuvat erilaisista laitteista ja mekaniikasta. Erilaisia laitteita ovat esimerkiksi ohjaintietokone, virtalähde, tai mittalaitteet. Laitteet voidaan jakaa kahteen kategoriaan:

- Kalibroinnin piiriin kuuluvat laitteet, joille valmistaja on määrittänyt ohjeellisen kalibrointi-intervallin ja hyväksyttävän tavan, jolla kalibrointi on suoritettava. Yleensä tällaiset laitekalibroinnit suorittaa ulkoinen laitekalibrointeihin erikoistunut palvelun tarjoaja. Palvelun tarjoaja voi olla esimerkiksi kyseisen laitteen valmistaja.

- Kalibroinnin piiriin kuulumattomat laitteet, joita ei ole tarpeen kalibroida. Tähän kategoriaan kuuluvat tyypillisesti laitteet, joilla ei suoriteta mittauksia, kuten ohjaintietokone tai useat virtalähteet.

Laitekalibrointien hallintaan suunniteltiin ja toteutettiin oma käyttöliittymänäyttö ja toiminnallisuus, joka ylläpitää tietoja kaikista kalibrointitapahtumista sekä luo automaattisesti uuden kalibrointitapahtuman, kun edellinen odottava kalibrointi on kuitattu suoritetuksi. Toimenpide suoritetaan aina käyttöliittymän avauksen yhteydessä. Ohjelma hakee kaikki tietokannassa olevat laitteet ja tutkii, onko niillä odottavaa avointa kalibrointitapahtumaa. Jos avointa kalibrointitapahtumaa ei löydy laitteelle löydy, ohjelma lukee kyseisen laitteen tiedoista kalibrointi-intervallin ja luo seuraavan odottavan kalibrointitapahtuman.

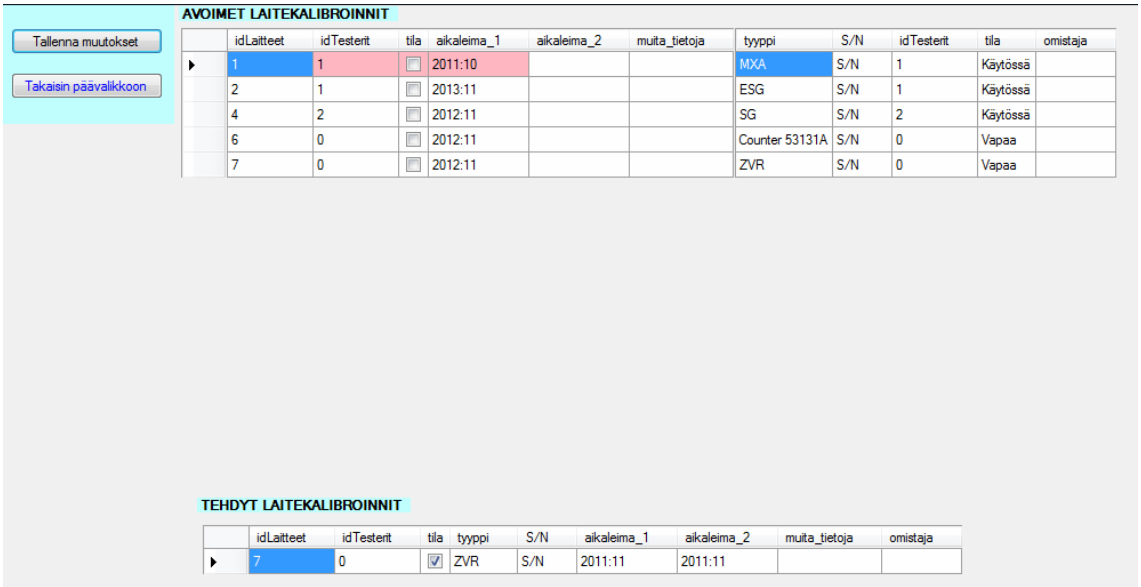
```
// Luodaan uudet kalibrointitapahtumat
int i3 = 1;
string aikaleima_1;
while (i3 < i)
{

    aikaleima_1 = DateTime.Now.AddMonths(kalibrointiväliArray[i3]).ToString("yyyy:MM");
    if (i_ok[i3] == "ok")
    { // ei tehdä mitään
    }
    else
    {
        sql = @"INSERT INTO laitekalibroinnit (idLaitteet,idTesterit,tila,aikaleima_1,aikaleima_2,muita_tietoja) VALUES("
            + idLaitteetArray[i3] + "," + idtesteritArray[i3] + ",FALSE,'" + aikaleima_1 + "','' ,' ')";
        MessageBox.Show(sql);
        cmd = new MySqlCommand(sql, conn);
        cmd.ExecuteNonQuery();
    }
    i3++;
}
```

Yllä on koodiesimerkki uusien kalibrointitapahtumien luomisesta. Laitteiden kalibrointitilanne on tallennettu i_ok-taulukkomuuttujaan. Uusi kalibrointitapahtuma luodaan vain sellaisille laitteille, joilla taulukkomuuttujan arvo on ”nok”.

Laitekalibrointinäkymä suunniteltiin pääasiallisesti kalibrointityötä ohjaavien ja suorittavien henkilöiden käyttöön. Näkymän avulla käyttäjän on helppo seurata

avoimia ja tehtyjä kalibrointitapahtumia, suunnitella tulevien kalibrointien tarvetta ja ajankohtaa tai kirjata tehtyjä tapahtumia järjestelmään. Ohjelma näyttää avoimet ja suljetut kalibrointitapahtumat omissa dynaamisesti luotavissa ruudukoissa (kuva 20).



KUVA 20. Avoimet ja tehdyt kalibroinnit näytetään erillisissä ruudukoissa. Myöhässä oleva kalibrointi indikoidaan punaisen taustavärin avulla

Suoritetun kalibrointitapahtuman voi yksinkertaisimmillaan kirjata tehdyksi klikkaamalla ensin hiiren avulla tilasarakkeessa olevaa ruutua ja seuraavaksi tallenna muutokset -nappia. Ohjelma lisää automaattisesti tallenteeseen suorituspäivämäärän ja luo kyseiselle laitteelle uuden huoltotapahtuman aiemmin kuvatulla tavalla.

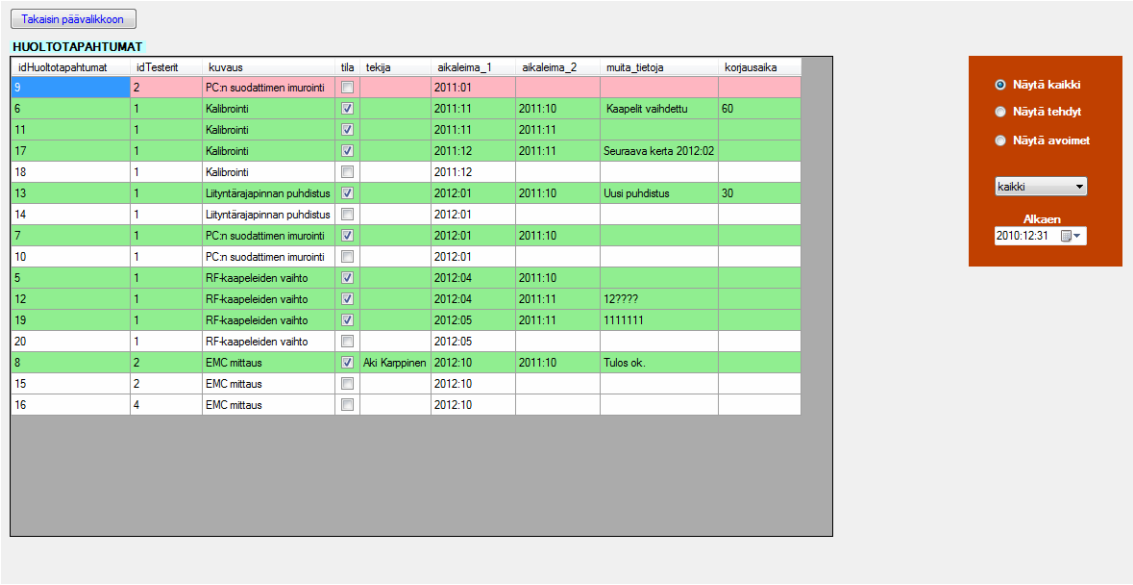
```
private void dataGridView2_CellValueChanged(object sender, DataGridViewCellEventArgs e)
{
    aikaleima = yht.Aikaleima();
    if (dataGridView2.Rows[dataGridView2.CurrentCellAddress.Y].Cells[3].Value.ToString() == "True")
    {
        dataGridView2.Rows[dataGridView2.CurrentCellAddress.Y].Cells[5].Value = aikaleima;
    }
    if (dataGridView2.Rows[dataGridView2.CurrentCellAddress.Y].Cells[3].Value.ToString() == "False")
    {
        dataGridView2.Rows[dataGridView2.CurrentCellAddress.Y].Cells[5].Value = "";
    }
}
```

Yllä on koodiesimerkki tapahtumankäsittelijästä, joka suoritetaan aina, kun käyttäjä tekee muutoksen johonkin ruudukon soluun. Metodissa tutkitaan tila-sarakkeen boolean-muuttujan arvo, ja sen perusteella joko asetetaan kyseiselle riville päivämäärä suoritetun kalibrointitapahtuman merkiksi, tai tyhjennetään

kyseinen solu. Jos muuttujan arvo on ”true”, eli ruutu on merkitty, tulostetaan vastaavan rivillä olevaan aikaleima_1-soluuun tapahtumahetken päivämäärä.

Testilaitteistojen huoltotapahtumien keskitettyyn tarkastelemiseen suunniteltiin yksinkertainen käyttöliittymänäyttö, joka on pääasiallisesti tarkoitettu henkilöille, jotka toimivat organisaatiossa huoltotyön johtotehtävissä. Toteutuksessa pyrittiin helppokäyttöisyyteen ja selkeyteen. Erityisesti pyrittiin huomioimaan se, että vuosien saatossa tietokantaan todennäköisesti kertyy merkittävä määrä huoltotoimitietoja. Käyttäjän tulisi vaivattomasti pystyä seulomaan haluamansa tiedot esiin myös suuresta tietomäärästä.

Ohjelma tulostaa näytölle käyttäjän tarkasteltavaksi kaikki järjestelmässä olevat huoltotapahtumat, niihin liittyvät parametritiedot ja ohjauspaneelin, jonka avulla käyttäjä voi rajata haettavia tietoja (kuva 21).



KUVA 21. Ohjauspaneelin hakukriteerien mukaiset huoltotapahtumat tulostetaan näytöllä olevaan ruudukkoon. Myöhässä oleva huoltotapahtuma indikoidaan punaisen taustavärin avulla

Ohjauspaneelissa on 3 radiopainiketta, valintaruutu ja päivämäärävalitsin. Radiopainikkeiden avulla voidaan asettaa huoltotapahtumien tilapalametri. Tilapalametrin arvo voi olla avoin, tehty tai kaikki. Valintaruudusta voidaan asettaa, haetaanko kaikkien testilaitteistojen huoltotapahtumia, vai vain jonkun

tietyn. Päivämäärävalitsimen avulla asetetaan päivämäärä, josta alkaen tietoja haetaan.

6 KEHITYSTYÖN TULOKSET

Tässä työssä kehitettiin sähköinen huoltolokijärjestelmä elektromekaanisten tuotteiden volyymituotannon eri vaiheissa käytettävien testauslaitteistojen ja laitteiden säännönmukaisen ylläpitotoiminnan hallinnointiin, seurantaan ja tulosten taltioimiseen. Ylläpitotoiminnalla käsitetään tässä yhteydessä edellä mainittujen kohteiden ennalta määritettyjä huoltotoimia ja kalibrointeja sekä ennalta suunnittelemattomia korjaustoimia.

Järjestelmä suunniteltiin elektromekaanisten tuotteiden valmistustoimintaa harjoittavien yritysten käyttöön. Koska työlle ei ollut varsinaista tilaajaa tai pilottiasiakasta, tarkennettiin vaatimusmäärittelyä tutkimalla vastaavia markkinoilta löytyviä kaupallisia ratkaisuja. Soveltuvuuden arviointiin käytettiin internethakupalveluiden kautta vapaasti saatavilla olevaa esittelymateriaalia.

Tutkimuksessa löydettiin useita kaupallisia vaihtoehtoja, joiden arvioitiin soveltuvan haluttuun käyttötarpeeseen, mutta niiden kaikkien todettiin kuitenkin olevan suunniteltu pääasiallisesti useiden eri teollisuudenalojen omaisuudenhallintaan ja valmistustuotannon ohjaukseen. Vaikka esittelymateriaalien pohjalta ei suoranaisesti ollut mahdollista määrittää tarkkoja hintatietoja, voitiin niistä kuitenkin päätellä kaupallisten järjestelmien yleisesti ottaen olevan varsin kalliita. Hinta-arviot vaihtelivat muutamista tuhansista euroista jopa useisiin kymmeniin tuhansiin euroihin. Useissa tapauksissa hinta ei ollut kertasuoritteinen, vaan sille määräytyi yksilöllinen toistuva vuosikustannus asiakkaan tarvitseman palvelun laajuuden mukaisesti.

Tutkimuksen pohjalta todettiin kaupallisten ratkaisujen olevan yleisesti ottaen laajoja ja kattavia. Niiden arvioitiin kuitenkin soveltuvan parhaiten keskisuurten tai suurten yritysten käyttöön jotka pääasiallisesti käyttävät tämäntyyppisiä järjestelmiä omaisuuden hallinta, tuotannonohjaus ja valmistusmateriaalin hallintakäyttöön. Edellisen perusteella potentiaaliseksi kohderyhmäksi kehitettävälle järjestelmälle valittiin pienet kotimaiset yritykset, joilla on suppea organisaatio ja joiden osaaminen keskittyy pääasiallisesti oman valmistustoiminnan avainosaamisalueisiin. Yrityksillä on kuitenkin tarve täyttää

sekä asiakkaiden, että ISO9001-standarin asettamat laatuvaatimukset ja pyrkiä valmistuksen tukitoimintojen osalla toimimaan kustannustehokkaasti.

Huoltolokijärjestelmän kehitystyössä tavoiteltiin helppokäyttöisyyttä ja vähäistä ylläpitotarvetta. Käyttöliittymäsuunnittelussa tavoiteltiin lisäksi selkeyttä ja syötevirheiden ennalta ehkäisemistä mahdollisimman kattavasti. Työkalu toteutettiin kauttaaltaan suomenkielisenä ja kahtena erillisenä asennettavana ohjelmana, asiakasohjelma.exe ja ylläpito-ohjelma.exe.

Asiakasohjelman käyttöympäristönä toimii testilaitteiston ohjaintietokone ja sen pääasiallinen käyttäjäryhmä on ylläpitotyön suorittajat. Asiakasohjelman tuottamia palveluita ovat

- huoltosuunnitelman näyttäminen
- huoltohistorian näyttäminen
- laitekokoonpano- ja tilatiedon näyttäminen
- käyttöaikatiedon taltioiminen
- korjaustöiden raportoiminen
- tehtyjen huoltotapahtumien kirjaaminen
- uusien huoltotapahtumien automaattinen luominen huoltosuunnitelman mukaisesti.

Jokainen yksittäinen ohjelma yksilöidään tietokantaliitoksen avulla juuri siihen testilaitteistoon, johon se on asennettu. Ohjelma linkitetään käyttöönoton yhteydessä kyseisen testilaitteistotunnuksen tietoihin ja liitos tallennetaan tietokoneen muistissa olevaan sisäiseen tietokantaan. Tietokantaliitoksen muodostamisen jälkeen ohjelman käyttö ei vaadi erillistä sisään kirjautumista ja yksilölliset testauslaitteistokohtaiset tiedot näytetään automaattisesti. Tämän

katsottiin tuovan merkittävää etua ylläpito- ja korjaustyön suorittajille, joiden työkuvan luonne edellyttää nopeaa ja helppoa tietoihin käsiksi pääsyä.

Ylläpito-ohjelman käyttäjiä ovat testilaitteistojen ylläpitotyön suunnittelusta, hallinnoinnista, laitehallinnasta ja laitekalibrointityöstä vastaavat henkilöt. Ohjelma asennetaan pääasiallisesti käyttäjän henkilökohtaiselle tietokoneelle. Vaihtoehtoisesti ohjelma voidaan asentaa myös yhteiskäyttöiseen tietokoneeseen. Väärinkäytön estämiseksi käyttäjille luodaan yksilölliset tunnukset, salasanat ja käyttöoikeudet. Ylläpito-ohjelman tuottamia palveluita ovat

- laitetietojen syöttäminen
- testilaitteistojen luominen (laitteen liittäminen laitteistoon)
- testilaitteistokohtaisten huoltosuunnitelmien luominen
- käyttäjätunnusten, salasanojen sekä niihin liittyvien käyttöoikeuksien luominen ja poistaminen
- salasanan vaihtaminen
- huoltotapahtumien näyttäminen
- kalibrointisuunnitelman näyttäminen
- suoritettujen kalibrointitapahtumien kirjaaminen
- uusien kalibrointitapahtumien automaattinen luominen
- käyttöaika- ja tilatiedon näyttäminen
- tietokannan varmuuskopioiminen ja palauttaminen.

Asiakas voi luvun alussa kuvattujen käyttötarkoitusten lisäksi saavuttaa lisäarvoa omassa toiminnassaan järjestelmään lisättyjen tai alkuperäiseen suunnitelmaan nähden paranneltujen toiminnallisuuksien avulla.

Järjestelmään lisättiin kehitystyön aikana toiminnallisuus, jonka avulla käyttäjä voi seurata hallinnoinnin piirissä olevien testilaitteistojen tila- ja käyttöaikatietoja. Tilatiedot päivittyvät järjestelmään reaaliajassa. Käyttöaikatietoja kerätään testilaitteistojen ohjaintietokoneiden kovalevyille vuorokaudenmittaisissa jaksoissa. Vuorokauden vaihtuessa kerätyt käyttöaikatiedot siirretään palvelimella sijaitsevaan tietokantaan. Testilaitteistojen käyttöaikatiedot ovat tarkasteltavissa järjestelmän kautta aina edellisestä vuorokaudesta alkaen taaksepäin.

Toinen järjestelmään lisätty toiminnallisuus on laitteiden ja laitteistojen yksilöinti. Järjestelmä luo automaattisesti edellä mainituille kohteille yksilölliset tunnisteet ja tietokentät, joita yritys voi hyödyntää kohteiden hallinnointiin ja paikantamiseen. Tunnisteet voidaan tarvittaessa tarroittaa kohteisiin, jos muita yksilöllisiä tunnistetietoja ei ole saatavissa tai ne ovat vaikeasti havaittavissa. Yhtenäisen merkintätavan ja yksilöllisten tunnisteiden avulla kohteiden tiedot löytyvät järjestelmästä vaivattomasti ja kohteet on helppo paikantaa myös tehdassalista. Tästä on hyötyä esimerkiksi, jos yrityksellä on käytössä ISO 9001 -sertifikaatin vaatimusten mukainen laadunhallintajärjestelmä ja se joutuu auditoinnin yhteydessä todentamaan kalibroinnin piirissä olevien laitteiden kalibroinnin tilan ja fyysisen sijainnin. Laitteiden nopea ja helppo paikantaminen on eduksi myös silloin, jos yrityksen on tarvetta suorittaa omaisuudenhallintaan liittyvä API (Asset Physical Inventory). Veroviranomainen voi vaatia yritystä todentamaan verotuksen piirissä olevan omaisuuden sijainti ja käyttötarkoitus verotusviranomaisen edustajalle.

Kolmas järjestelmään lisätty toiminnallisuus on mahdollisuus seurata laitteiden elinkaaren tilaa ja ympäristöjärjestelmän (ISO 14001) mukaisuutta laitehallintanäkymässä. Jokaisella laitteella on tätä tarkoitusta varten omat parametrikentät, joihin kyseiset tiedot voidaan asettaa. Yritys voi esimerkiksi sisällyttää järjestelmän osaksi omaa sertifioitua ympäristöjärjestelmää ja

hyödyntää tätä toiminnallisuutta esimerkiksi laitteiden elinkaarenjälkeisen kierrätyksen apuna erottelemaan kierrätyksen kannalta ongelmattomat laitteet ongelmattomista.

7 POHDINTA

Tämä opinnäytetyö on ollut yhden miehen projekti alusta asti. Työn aiheen keksin omien työelämästä saamieni kokemusten pohjalta. Olen työskennellyt telekommunikaatioalalla erilaisissa tehtävissä yli kymmenen vuoden ajan ja tutustunut sitä kautta muun muassa erilaisiin testausteknologioihin, testauksen suunnitteluun, testauslaitteisiin sekä muihin alaan liittyviin asioihin. Ajatus opinnäytetyön tekemisestä oman aiheen pohjalta jalostui pikkuhiljaa opintojen edetessä, kun mietin sitä, kuinka opintojen varrella opittuja uusia tietoja ja taitoja voisi soveltaa koettujen käytänteiden ja rutiinien kehittämiseen jatkuvan parantamisen hengessä.

Tavoiteasetannan aikana pyrin miettimään työn laajuuden mahdollisimman kattavaksi opintojeni painopiste ja käytettävissä oleva aika huomioiden siten, että toteutus olisi riittävän haastava, mutta kuitenkin toteutettavissa oleva eheä kokonaisuus. Työssä tuli olla mahdollisimman todenmukainen asiakastarve, johon lopputuotos esittää ratkaisun. Järjestelmän koostumusta ja toiminnallisuutta määritellessäni jaoin ensimmäisessä vaiheessa kaikki palaset kahteen kategoriaan, pakollisiin ja valinnaisiin. Pakolliseen kategoriaan kuuluivat sellaiset järjestelmäosat ja toiminnallisuus, joita ilman järjestelmäkokonaisuuden vaatimia perustarpeita ei olisi saatu toteutettua. Tällaisia perustarpeita ovat esimerkiksi tietokanta ja käyttöliittymätoiminnot, joiden avulla käyttäjä kykenee syöttämään järjestelmään tietoja tai tarkastelemaan niitä. Valinnaiseen kategoriaan kuuluivat sellaiset järjestelmäosat ja toiminnallisuus, joiden puuttumisesta huolimatta perustarpeet tulisi tyydytettyä. Tähän kategoriaan kuului esimerkiksi kehitysvaiheessa ajanpuutteen vuoksi pois jättämäni etäkäyttöliittymä, jota oli tarkoitus käyttää tietojentarkasteluun internetselaimella. Etäkäyttöliittymä ei ollut pakollinen, koska sama toiminnallisuus kyettiin sisällyttämään vaihtoehtoisesti ylläpito-ohjelmaan.

Kehitystyön päätin toteuttaa 4-vaiheisena ohjelmistokehitysprojektina. Määrittelyvaiheen alusta lopetusvaiheen päättymiseen saakka suunniteltu kesto oli 11 viikkoa. Ohjelmistoprojektin luonteelle on tyypillistä se, että alun perin

ohjelmistolle asetetut vaatimukset lisääntyvät tai tarkentuvat kehitystyön edetessä. Yksi syy tähän on se, että tämän kokoluokan ohjelmisto on niin kompleksinen kokonaisuus monine rajapintoineen ja riippuvuuksineen, että kaikkia asioita ei kyetä mitenkään huomioimaan etukäteen. Toinen syy on se, että myös asiakkaan suunnasta tulevat vaatimukset muuttuvat työn edetessä sitä mukaa, kun asiakas näkee valmista toiminnallisuutta. Näin kävi myös tässä projektissa, vaikka toimin itse sekä asiakkaan, että toteuttajan rooleissa. Tämän kaltaiselle yhden miehen projektille ei tietääkseni ole virallista termiä tai menetelmämääritystä. Itse kutsuisin menetelmää termillä "ketterä+". Perusteluna se, että asiakas on läsnä koko kehitystyön ajan antaen välitöntä palautetta toteutukseen. Koko toteutusvaihe etenee ikään kuin äärimmäisen lyhyiden iteraatiokierrosten varassa.

Kehitystyön tuloksiin olen kaiken kaikkiaan tyytyväinen. Sain järjestelmän valmiiksi asetettujen tavoitteiden ja suunnitellun aikataulun mukaisesti. Lopputuote on mielestäni alun perin suunnittelemaani monipuolisempi ja asetettujen vaatimusten mukainen. Lopputestausta ja toiminnallisuuden hiomista olisi toki voinut jatkaa loputtomiin, muuta näissä olosuhteissa siitä ei tämän opinnäytetyön kannalta olisi ollut merkittävää hyötyä. Tämän asian hahmottamisessa projektin loppumetreillä tarvittiin myös ohjaajan näkemystä. Itse olin vielä toteutusvaiheen umpeutuessa niin vahvasti asiakkaan roolissa kiinni, että vaikka tavoite saattaa työ loppuun suunnitellun mukaisesti oli kristallin kirkas, asiakkaan äänen johdattamana olisin jatkanut ohjelman toiminnallisuuden hiomista ja työn valmistuminen olisi siirtynyt todennäköisesti myöhempään ajankohtaan.

Järjestelmää voisi edelleen kehittää paremmaksi monin tavoin. Alla muutamia kehitysideoita, joita ei tämän työn puitteissa ollut mahdollista toteuttaa.

Asiakasohjelmaa suoritetaan palveluna aina, kun testilaitteiston ohjaintietokone on käynnissä. Ohjelmaan on lisäksi rakennettu niin sanottu keinotekoinen ajantaju, jonka avulla ohjelma kykenee asettamaan aikaleimat tallenteisiin ja vertaamaan tietokannassa olevia aikaleimoja nykyhetkeen. Tätä toiminnallisuutta voisi kehittää edelleen siten, että ohjelma valvoo

automaattisesti huoltotapahtumien ja laitekalibrointien määräpäiviä ja lähettää tietyn varoajan puitteissa ennalta määrätylle jakelulle tiedotteen sähköpostin välityksellä, mikäli määräaika uhkaa umpeutua.

Järjestelmän avulla voi luoda erilaisia dynaamisia näkymiä tietokannassa olevien tietojen pohjalta. Järjestelmään voisi lisätä toiminnallisuuden jonka avulla edellä mainittuja näyttöjä ja listauksia voisi tulostaa paperille tai vaihtoehtoisesti sähköisenä tallenteena halutussa formaatissa kovalevylle.

On todennäköistä, että järjestelmää käyttöön otettaessa yrityksellä on jo lukuisia olemassa olevia mittalaitteistoja ja laitteita sekä vastaava kirjanpito jossain formaatissa (Excell, Word, Xml tai vastaava). Kohteiden syöttäminen tietokantaa yksittäin on työlästä ja siihen kuluu paljon aikaa. Järjestelmään voisi lisätä tietojen syöttötoiminnon, jonka avulla tietojen massasyöttö olisi mahdollista suoraan sähköisestä tallenteesta.

Mikäli järjestelmän tuotteistamista harkittaisiin vakavasti, tulisi suorittaa asianmukainen koekäyttö todellisessa käyttöympäristössä. Tähän tarkoitukseen tulisi etsiä sopiva pilottiasiakas tai pilottiasiakkaita, joiden kanssa koekäytön ehdoista sovittaisiin. Koekäytön avulla saataisiin arvokasta tietoa järjestelmän toimivuudesta, mahdollisista ongelmista sekä ulkopuolisen asiakkaan näkemys kehityskohteista.

Työ on ollut haastava ja mielenkiintoinen. Sen eri vaiheiden aikana olen saanut edelleen kehittää omia taitojani useilla eri osa-alueilla aina ongelmanratkonnasta ajanhallintaan ja dokumentaatiosta ohjelmointiin.

LÄHTEET

1. SFS-EN ISO 9001. 2001. Laadunhallintajärjestelmät. Vaatimukset. Helsinki: Suomen Standardoimisliitto SFS.

2. SFS. 2011. Laadunhallintajärjestelmän luominen. Saatavissa: http://www.sfs.fi/iso9000/laadunhallinta/ . Hakupäivä 01.12.2011.

3. SFS-EN ISO 14001. 1996. Ympäristöjärjestelmät. Spesifikaatio ja ohjeita sen käyttämiseksi. Helsinki: Suomen Standardoimisliitto SFS.

4. SFS. 2011. Ympäristöjärjestelmä. Saatavissa: http://www.sfs.fi/iso14000/ymparistojarjestelma/.Hakupäivä 02.12.2011.

5. Oracle Corporation. 2011. About MySql. Saatavissa: http://www.mysql.com/about/. Hakupäivä 02.12.2011.

6. Apache Software Foundation. 2011. About Apache. Saatavaissa: http://httpd.apache.org/ABOUT_APACHE.html. Hakupäivä 03.12.2011.

7. Bruno Pagès. 2011. Bouml. Saatavissa: http://bouml.free.fr/. Hakupäivä 03.12.2011.

8. Wikipedia Foundation Inc. 2011. Microsoft Visual Studio. Saatavissa: http://en.wikipedia.org/wiki/Microsoft_Visual_Studio. Hakupäivä 03.12.2011.

9. SFS. 2011. Ympäristöasioiden hallinta. Saatavissa: http://www.sfs.fi/files//iso14000esite.pdf. Hakupäivä 06.12.2011.

HUOLTOTAPAHTUMAT-TAULUN MYSQL-SKRIPTI LIITE 1

```
-- -----------------------------------------------------
-- Table `huoltolokitietokanta`.`Huoltotapahtumat`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `huoltolokitietokanta`.`Huoltotapahtumat` (
  `idHuoltotapahtumat` INT NOT NULL AUTO_INCREMENT ,
  `idTesterit` INT NOT NULL ,
  `idvHuoltotoimet` INT NOT NULL ,
  `tila` TINYINT(1)  NOT NULL ,
  `tekija` VARCHAR(45) NULL ,
  `aikaleima_1` VARCHAR(20) NOT NULL ,
  `aikaleima_2` VARCHAR(20) NULL ,
  `muita_tietoja` VARCHAR(120) NULL ,
  `korjausaika` INT NULL ,
  PRIMARY KEY (`idHuoltotapahtumat`, `idTesterit`, `idvHuoltotoimet`) ,
  INDEX `testeri` (`idTesterit` ASC) ,
  INDEX `Huoltoimi` (`idvHuoltotoimet` ASC) ,
  UNIQUE INDEX `idHuoltotapahtumat_UNIQUE` (`idHuoltotapahtumat` ASC) ,
  CONSTRAINT `testeri`
  FOREIGN KEY (`idTesterit` )
  REFERENCES `huoltolokitietokanta`.`Testerit` (`idTesterit` )
  ON DELETE RESTRICT
  ON UPDATE CASCADE,
  CONSTRAINT `Huoltoimi`
  FOREIGN KEY (`idvHuoltotoimet` )
  REFERENCES `huoltolokitietokanta`.`Valitut_Huoltotoimet` (`idVHuoltotoimet` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

ASIAKASOHJELMAN LOHKOKAAVIO **LIITE 2**